Honeywell

# TIME SHARING TEXT EDITOR

SERIES 60(LEVEL 66)/6000

SOFTWARE

Honeywell

# TIME SHARING TEXT EDITOR

SERIES 60(LEVEL 66)/6000

SUBJECT:

Description and Use of the Text Editor.

SPECIAL INSTRUCTIONS:

This manual replaces _Time Sharing Text Editor_, Order Number BR40 for Series 6000 System users. Order Number BR40 must be used by Series 600 System users and by Series 6000 System users who are on prior software releases.

SOFTWARE SUPPORTED:

Series 60 Level 66 Software Release 2
Series 6000 Software Release H

DATE:

February 1975

ORDER NUMBER:

DD18, Rev.0

## PREFACE

This manual describes the Text Editor feature of the Honeywell Series 60 Level 66 and Series 6000 System and is intended for use by the terminal operator at a remote site. No programming knowledge is required of the terminal operator. The Series 60 Level 66 is hereafter referred to as Series 60. The information in this manual refers to both the Series 6000 and the Series 60, unless otherwise specifically stated.

File No.: 1733,1P33

## CONTENTS

CONTENTS (cont)

## SECTION I

## INTRODUCTION

Text Editor, a major feature of the Time Sharing System, is a means by which files consisting of text may be entered, edited, stored, retrieved, and formatted. Text Editor consists of two subsystems--EDITOR and RUNOFF.

The EDITOR subsystem permits such text as correspondence, mailing lists, inventories and records to be entered and readily modified by additions, deletions, and corrections. The EDITOR subsystem is described in Section III.

The RUNOFF subsystem, in conjunction with EDITOR, permits the user to specify the format in which text is to be reproduced. RUNOFF features include the ability to specify the length of each line of text, the number of lines on a page, and the size of page margins. All RUNOFF features are described in Section IV.

Communication with the Time Sharing System and the computer is by means of a terminal. The terminal used, of course, is determined by the user's site. Descriptions of terminal use in this manual assume a typical terminal. The user should read the operation manual of the terminal he is to use to become familiar with its operation. <u>TSS General Information Manual</u>, contains description of operations of several terminals that could be used with the Time Sharing System.

Commands, format control words, and text are depicted in upper case in this manual as a matter of expediency. If the terminal in use permits, the choice of upper or lower case for text is a user's prerogative. As for commands and format control words, the Time Sharing System, the EDITOR subsystem, and the RUNOFF subsystem recognize all such, regardless of case.

## SECTION II

## TEXT EDITOR AND THE TIME SHARING SYSTEM

### TEXT EDITOR

The user of Text Editor must work within the confines of the Time Sharing System to communicate with the computer and perform nonediting functions on the files. The following paragraphs contain both background and essential information concerning the Time Sharing System and its relation to the Text Editor. The user should refer to the TSS General Information Manual for details concerning the Time Sharing System.

### TIME SHARING

Time sharing is the term used to describe the system by which a user appears to be operating his terminal as if he has exclusive use of the computer. The user can enter text files, modify and delete parts, and then store and retrieve the files at his convenience. If errors are made in the way the Time Sharing System is used, he is informed by error messages printed at his terminal.

### CONNECTING TERMINAL TO THE COMPUTER

In order to connect with the computer from a terminal, proceed as follows:

1. Turn unit on and obtain a dial tone.
2. Dial one of the numbers at the Time Sharing Center.

When the connection is made, a high-pitched tone is received, then no tone at all, and the terminal prints out an indication that the computer is available and that communications with the computer, through the terminal, can now be made.

### LOG-ON PROCEDURE

With the terminal connected to the computer, the system initiates a "log-on" procedure. During this procedure the terminal asks for information and a proper response must be made, each response followed by a carriage return (achieved by depressing the RETURN key). First, the terminal asks for a user's identification. This is a string of characters that is assigned to uniquely identify the user to the computer for the purpose of identifying his program and accounting for the user's charges.

The terminal next asks for a password. The area on which the password is printed is scored over by the terminal to make the password illegible. The purpose of this password is to ensure the computer that it is "talking" to the legitimate user and not someone else using his identification. The password is his protection against unauthorized use of his user identification. If the password is valid, the system prints an initial asterisk to indicate readiness to accept system commands or subsystem names as input from the terminal.

A typical log-on sequence follows:

```
HIS SERIES 6000 ON 05/10/77 AT 9.183 CHANNEL 0012
USER ID--DOE
PASSWORD
ABCDEFGHIJKL
*
```

## LOG-OFF PROCEDURE

If the user types the control command BYE, the Time Sharing System gives the user a summary of the amount of time and resources used this run and the total amount of the user's resources used to date. The Time Sharing System then logs the user off by disconnecting the terminal.

## AUTOMATIC DISCONNECTION FROM THE TIME SHARING SYSTEM

The user is automatically disconnected from the Time Sharing System for any of the following reasons:

- If he responds twice with invalid user identifications. The terminal replies after the first invalid use with the message ILLEGAL ID--RETYPE--. If the user responds with an invalid user identification a second time, he is disconnected.
- If he responds twice with invalid passwords. The terminal replies after the first invalid use with the message ILLEGAL PASSWORD--RETYPE--. If the user responds with an invalid password a second time, he is disconnected.
- If more than one minute passes without a user response to user identification or password request.
- If he leaves the terminal in an idle state for over ten minutes.
- If his resources are overdrawn by more than 10 per cent. The message, RESOURCES EXHAUSTED. CANNOT ACCEPT YOU, is printed by the terminal before disconnection takes place.

## EXAMPLE OF TERMINAL OPERATION AND PROCEDURES

The following elementary example illustrates steps in terminal operation and procedures required for entering text and saving the file.

```
HIS SERIES 6000                          } Acknowledgment of
ON 01/15/77 AT 9.891 CHANNEL 0020        } terminal connection
                                         } to Time Sharing System.

USER ID --DOE                            }
PASSWORD--                               }
XXXXXXXXXXXX                             }
*EDITOR                                  } Log-on procedure.
-BUILD
ENTER
*text----
*text----
*Carriage return upon completion of text
-SAVE  filename                            Request to save file
DATA SAVED  filename                       File saved.
BYE
**RESOURCES USED $ 0.32, USED            } Log-off procedure and
TO DATE $  35.00=10%                     } accounting.
**TIME SHARING OFF AT 10.006             }
ON 01/15/77                              }
```

The EDITOR can be entered from another subsystem when the user issues a (-) followed by a recognizable Text Editor command. For example:

```
USER ID--DOE
PASSWORD--
XXXXXXXXXXXX
*FORTRAN OLD FORT1
*-PRINT                        User requests the EDITOR to print first
                               line of the file.
THIS IS THE ABC PROGRAM
*-REPLACEVS:/ABC/:/XYZ/        Replace string ABC with XYZ and verify (V).
THIS IS THE XYZ PROGRAM        The EDITOR prints the line as changed.

*                              System returns to the FORTRAN build mode.
```

## ERROR MESSAGES AND THE HELP SUBSYSTEM

Some user errors cause messages to appear at the terminal. These messages are prefixed by an error code number. The messages are intended to be self-explanatory and are brief. If a fuller explanation of the error is required, the time sharing HELP subsystem can be called.

The following example of the use of the HELP subsystem illustrates a spelling error made by the user during log-on.

```
        *RANOFF
     ┌──009-SYSTEM UNKNOWN
┌────┼──*HELP
│    │
│ ┌──┼──PLEASE ENTER MESSAGE NUMBER-9
│ │  │
│ │  │  THE REQUESTED SUBSYSTEM IS UNKNOWN TO TSS OR IS NOT
│ │  │  INCLUDED IN THE SYSTEM FOR THIS INSTALLATION. CHECK
│ │  │  THE NAME FOR SPELLING TOO.
│ │  │
│ │  │  *
│ │  │
│ │  │
│ │  └──Error message
│ └─────Typed out by HELP.  Reply with error code number
└───────Explanation typed by HELP
```

## PRINTING OR PUNCHING COPIES OF A FILE

If the user desires, the contents of a text file can be printed or punched on cards at the computer's operational site. The printing process provides a means of copying lengthy files quickly if a high-speed printer is available at the site. The punching process generates a card deck copy, at the site, of a file which may be convenient for duplication or storage.

The printing process is accomplished through BPRINT and the punching process through BPUNCH, two subsystems of the Time Sharing System. The following example illustrates the use of either of these subsystems.

```
*BPRINT} filename
 BPUNCH}
$ IDENT? account number,identification
LABELS? ASIS  (This indicates the file is a text file and
               does not contain line numbers.)
TAB CHARACTERS AND SETTINGS? t ,c  ,c  ,c  ...; t ,c  ,c  ,c  ...
                             (null, if no tabs are required)

       t = tab character (see RUNOFF, Section IV)
       c = column numbers
```

The user can place the label and tab information in the first line of his file, in which case, only the $ IDENT information is requested after a BPRINT or BPUNCH typein. The format of this first line is as follows:

```
##ASIS  t ,c  ,c  ;t ,c  ,c  ...
```

Refer to the manual, TSS Terminal/Batch Interface for further information on the use of BPRINT and BPUNCH.

## HOLD/SEND SUBSYSTEMS

The Time Sharing System issues information and warning messages that may appear at the user's terminal at any time. The injection of these messages into a printout of a file or into a paper tape create activity is an annoyance which can be eliminated by use of the HOLD/SEND subsystems of the Time Sharing System.

The HOLD response prevents Time Sharing System messages from appearing at the terminal until such time as the user gives a SEND. The last message from the Time Sharing System that occurred while the terminal was in the HOLD mode appears, once the SEND response is given.

# SECTION III

# EDITOR SUBSYSTEM

## EDITOR SUBSYSTEM FUNCTIONS

The EDITOR subsystem consists of functions that permit the user to perform the following:

1. Build a text file.
2. Append to an existing text file.
3. Edit a text file by additions, deletions, or corrections.

A comprehensive set of editing commands are available for use by the EDITOR subsystem to perform these functions.

## ENTRY TO EDITOR SUBSYSTEM

Following the log-on procedure described in Section II, the user responds to the initial asterisk with EDITOR. A hyphen (-) then appears to indicate the availability of the EDITOR subsystem.

What action the EDITOR subsystem takes upon being called is dependent upon the file accessed. The file accessed can be in one of two possible conditions:

1. The file contains no text, as in the case of a new file to be built, or possibly no text exists in the file accessed by an OLD <u>file name</u> response.
2. Text exists in the file accessed by an OLD <u>file name</u> response.

Under the first condition, wherein no text exists, the EDITOR subsystem transmits the editing response, ENTER, to the terminal and calls upon the Time Sharing System data collector to issue an asterisk. The asterisk indicates that the system is in build mode and system commands or subsystem names are acceptable. The entry sequence is as follows:

```
*EDITOR NEW
 ENTER
*
```

Under condition two, wherein text does exist, the EDITOR subsystem accepts any editing command following the hyphen response. The entry sequence is as follows:

```
     *EDITOR OLD filename
     -(any editing command)
```

If the user desires to append data to the file _filename_, the editing command BUILD is entered and an ENTER and asterisk are transmitted to the terminal as in the first condition. The entry sequence is as follows:

```
     *EDITOR OLD filename
     -BUILD
     ENTER
     *
```

NOTE: In the first few examples shown, user entries are underscored, as a teaching aid. These underscores are not part of the file and do not appear with entries made at the terminal.

## BUILDING OR ADDING TO A FILE

After the message, ENTER, and the initial data collector asterisk, two methods can be used to build (create) or add to a file. Text can be entered from the terminal via the keyboard, or from paper tape if the terminal is equipped with a tape reader.

### Entering Text From Terminal Keyboard

At the keyboard, begin typing in the desired text. After each carriage return, the system types out an asterisk at the beginning of each new line. This asterisk does not appear in the line of text in any printout of the file.

The following rules apply when entering text:

1. Text can be typed using both upper and lowercase letters if both are available on the terminal.

2. The desired text is typed immediately following the asterisk. All characters, including spaces, typed after the asterisk appear in the printout of the file.

```
*EDITOR NEW
ENTER
*THIS LINE IS TYPED WITHOUT LEADING SPACES.
*     THIS LINE CONTAINS 5 LEADING SPACES.
*(carriage return)
-PRINT;*
THIS LINE IS TYPED WITHOUT LEADING SPACES.
     THIS LINE CONTAINS 5 LEADING SPACES.

END OF FILE
```

3. To insert a blank line, use the space bar and then the carriage return.

   As shown in the example, a carriage return immediately following the asterisk terminates the text entry and produces the "-" response. At this point, editing or time sharing commands can be issued.

4. A carriage return is required at the end of every line of text entered and upon completion of text entry.

5. On a keyboard/display type terminal, the first character typed in replaces the asterisk. To terminate text entry and use an editing command, type two pound signs (##) and a blank following the asterisk.

6. Service functions recognizable with text entry are #AUTO, #TAPE, #LUCID, #RECOVER, and #ROLLBACK. Refer to the <u>TSS General Information Manual</u>.

## Line Numbered Files

Line numbers are not required by the EDITOR or RUNOFF subsystems, but line numbered files are required by most of the other time sharing subsystems. The user can employ the EDITOR and RUNOFF functions on line numbered files for later use under another subsystem. The user can supply one to eight numeric characters as the first entries for each line, or line numbers can be supplied automatically by the Time Sharing System by the use of the #AUTO command in the "BUILD" mode. #AUTO can be used as follows:

1. #AUTOMATIC

   Causes the automatic creation of line numbers by the system, at the point at which the automatic mode is entered (or re-entered), with line numbers initially starting at 010 and incrementing by 10 (or, on re-entry, resuming where the previous automatic numbering left off). These line numbers appear in the terminal copy, and are written in the file, just as though the user had typed them.

2. #AUTOMATIC n,m

   Causes the automatic creation of line numbers, as above, but starting with line number <u>n</u> and incrementing by <u>m</u>.

3. #AUTOMATIC ,m
   #AUTOMATIC n,

   Causes automatic creation of line numbers beginning at 10 and incrementing by m, or beginning at n and incrementing by 10 (on re-entry, the line numbering resumes where it left off).

Normally the line number is followed by a blank. Any nonblank, nonnumeric character affixed to the end of the command #AUTOMATIC causes the blank to be suppressed. For example: #AUTONB or #AUTOMATICX.

No commands are recognized while in the automatic mode. The automatic mode is cancelled by giving a carriage return immediately following the issuance of an asterisk and line number by the system. Upon leaving "#AUTO", return is to the EDITOR "BUILD" mode. The user may not use character delete (@) or line delete (CTRL X) to delete characters associated with the generated line number or its associated blank.

## Resequencing Line Numbered Files

The RESEQUENCE command can be used to resequence the line numbers of the current file. The RESEQUENCE command must be utilized while in the "edit" mode of the Text Editor.

The description of the RESEQUENCE command is in the TSS General Information Manual and repeated below for easy reference.

1. RESEQUENCE

   The line numbers of the current file are resequenced. The resequencing begins with line number 10 and continues in increments of 10. If BASIC is the selected subsystem, the file is resequenced and statement number references in the program are modified correspondingly (GOTO, GOSUB, IF, ON, Print USING). If FORTRAN or CARDIN was selected, statement number references are not affected.

2. RESEQUENCE n,m,x-y

   The line numbers of the current file are resequenced and modifications made according to the subsystem selection. The resequencing begins with line number n and continues in increments of m.

   x and y are specified only if partial resequencing is desired. x gives the starting point and y the ending point of resequencing, inclusive. A null x field (i.e., -y) specifies from beginning of file to line y, and a null y field (i.e., x-) specifies from line x to the end of file.

   In general, any blanks preceding a line number are stripped off. Unnumbered lines are accepted, except under the BASIC subsystem, and will have line numbers added, as implied or specified in the command. Care should be taken in resequencing concatenated BASIC files as line numbers are also statement numbers, and statement references, after resequencing, may become invalid.

3. RESEX n,m

Line numbers are inserted at the beginning of each line in the current file, regardless of whether or not line numbers already exist. The numbering begins with n and increments by m, or optionally, begins with 10 and increments by 10, if n,m are not specified. If the first character of the existing line is a numeric, a blank is inserted following the generated line number. If the first character of the existing line is not numeric, no blank is inserted.

4. RESE# n,m

Line numbers are inserted at the beginning of each line in the current file, even if line numbers already exist. This numbering begins with n and increments by m, or optionally begins with 10 and increments by 10 if n, m are not specified. If the first character of the existing line is a numeric, a pound sign (#) is inserted following the generated line number. If the first character of the existing line is not numeric, the pound sign is not inserted.

CAUTION: When resequencing, or performing a partial resequence, it is possible to produce files with line numbers out of order. This may be caused by incorrect parameters on partial resequence or when new line numbers exceed eight digits (in non BASIC files). When line numbers are too large, a warning is given. In either case, recovery may be made by resequencing the total file using a smaller beginning line number or a smaller increment.

## Entering Text From Paper Tape

A text file can be created on paper tape to be entered into the computer at a later time. To do this, put the terminal in LOCAL, feed enough tape through the tape drive to ensure that there are no unwanted characters, and type the text. A carriage return, line feed, and two rubouts must follow every line of text. An X-OFF must indicate completion of the text, followed by two rubout characters which provide a timing factor.

To use a prepared tape, enter the EDITOR subsystem, and type #TAPE following the initial asterisk. When the READY response appears, put the prepared tape in the terminal's tape reader and turn on the tape drive. The terminal must be in the online mode.

Input from the tape is accepted until the terminal operator stops the reader, the tape runs out or jams, or an X-OFF character is read from the tape. As the tape is being read, a copy of its contents is printed out on the terminal. When tape input is complete, the system looks for an X-OFF prior to transmitting a carriage return and printing an asterisk. At this time, additional text may be entered at the keyboard or a carriage return can be given to obtain the "-" response and allow editing or printing.

```
*EDITOR NEW
ENTER
*#TAPE
A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK. EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY,
THE COMPUTER PROCESSES AND PRODUCES INFORMATION
AS DIRECTED BY THE PROGRAM.

A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.
(X-OFF)
*(carriage return)
-
```

The #TAPE command can also be used to add text from paper tape to a text file that has been built in a current session at the terminal or has been previously saved (refer to the SAVE and OLD command descriptions in the TSS General Information Manual).

```
-BUILD
ENTER
*#TAPE
(Read from paper tape as described above.)
(X-OFF)
*(carriage return)
-
```

The #LUCID request is substituted for #TAPE for non-ASCII paper tape input.

A printout of the file shows text from paper tape appended to the original text.

Text from paper tape can be inserted into a file at any point in the file. Refer to the description of ENTER under "Responses From EDITOR" in this section.

## PROTECTING FILES

An automatic terminal disconnect, a computer or communication lines malfunction, or user simply forgetting to save a file before shutting down can cause the loss of input if the user is building or adding to a file. A large file requiring many hours or even days of typing input may be lost. The following paragraphs describe methods of preventing such losses.

The simplest way to ensure against loss from any condition except computer system malfunction is to save portions of the file at intervals while building. In this way, only the last unsaved portion of the file would be subject to loss. (See the following example.)

```
*EDITOR NEW
ENTER
*text ---------
*text ---------
*
*
*
*
*text ---------
*(carriage return)
-SAVE EXAM.1
DATA SAVED--EXAM.1
-
(At this point, you can use the editing commands to print or change the
file.  For each succeeding save, use the RESAVE function and specify the
original name.  If you wish to continue building this file, use the BUILD
command.)

-BUILD
ENTER
*

Request for editing function
Request for SAVE
Verification of SAVE
```

NOTE: The use of commands #RECOVER, #ROLLBACK, OLDP, and OLDP# can provide the user with additional means of file protection. Refer to the _TSS General Information Manual_ for details of use of these commands.

A paper tape of the file contents also provides a hard copy backup in case a file must be rebuilt.

This tape can be punched as you build the file from the keyboard. The tape will contain the asterisk printed by the system at the beginning of each line and any lines which were deleted or corrected while building. (See the following example.) If it is necessary to rebuild this file via tape, the rebuilt file must be edited.

```
*EDITOR NEW
ENTER
*#TAPE
READY
*HUMAN LANGUAGES ARE IMPRACTICAL FOR PREPARING
*COMPUTER PROGRAMS BECAUSE THESE LANGUAGES
*CONTAIN MANY AMBIGUITIES AND REDUNDANCIES;
*THE COMPUTER INTERPRETS LANGUAGE ABSOLUTELY
*LITERALLY. BY THE SAME TOKEN, MACHINE
*LANGUAGES ARE ALSO IMPRACTICAL BECAUSE THEY ARE
*DIFFICULT FOR PEOPLE TO USE. MOST PROGRAMMING
*LANGUAGES ARE COMPROMISES BETWEEN HUMAN AND
*MACHINE LANGUAGES.
(X-OFF)
*(carriage return)
-
```

To create a tape that does not require extensive editing, build a portion of the file, enter the editing function, give a PRINT command, and punch the tape as the file is being printed out. The following example illustrates this method.

```
*text
*text
*(carriage return)
-PRINT;*    (Do not type a carriage return until the tape drive has
            been turned on and the following done:
            (1)  To ensure a clean tape, repeat the rubout character
            until you have a tape leader long enough to be placed in
            the tape drive.
            (2)  Backspace the tape once so that the carriage return
            is wiped out by the last rubout character.
Type a carriage return.
```

The contents of the file is typed out while the tape is being punched. The message END OF FILE is punched into the tape. If the file must be rebuilt via this tape, be sure to delete this message.

## SEARCH POINTER CONVENTIONS

Each file upon entering EDITOR has a search pointer associated with it. This pointer is located at the beginning of the file until the first editing command is given and is backed up a specified number of lines or returned to the beginning of the file by the BACKUP command. The pointer always points to the beginning of a line, never to a point within the line. This allows several edit operations to be performed on the same line, so long as the operation does not move the search pointer.

The rules governing the movement of the search pointer are as follows:

1. The PRINT, INSERT, REPLACE, DELETE, FIND, CUT, COPY, and PASTE commands cause the search pointer to move forward toward the end of the file, unless the command affects only the line at which the pointer is already located (usually a command with no operand field).

2. Following the execution of any of the commands listed in rule 1., the pointer is located at the last line affected by the command.

3. The BUILD command positions the search pointer to the end of the file. Exiting from the BUILD repositions the search pointer to the beginning of the file.

4. The BACKUP command moves the search pointer backward to the beginning of the file or a specified number of lines from wherever it is located.

5. For commands involving a search operation--a string field is specified--the file is always searched starting at the current location of the search pointer; the search is terminated either by a successful comparison with the specified string field or by encountering the end of the file. In the latter case, the pointer must be backed up before any further editing operations may be performed.

NOTE: In the line mode, the search pointer can be moved forward or backward by the use of +n or -n with a search verb. "n" is the number lines to move forward (+) or backward (-).

If a given line has already been passed by the search pointer, the BACKUP command or a command with a -n mode must be used to backup the pointer to the line to be operated on.

The current position of the search pointer can always be determined by using a PRINT command with no operand field.

The position of the search pointer is also affected by use of the terminal "break" key to halt the file printout process. The position of the search pointer at the time the break key is depressed is dependent upon the system interfaces. If internal procedures have not been completed when the EDITOR subsystem is notified, the search pointer is positioned back at the last "-" response. If the internal procedures have already been completed prior to transmitting the - response to the terminal, then in all likelihood the search pointer position and command execution is as if the break has not occurred.

The following symbols are used in some examples illustrating the location of the search pointer:

◇Location of search pointer at the start of command execution.

▷Location of search pointer at the finish of command execution.

◁▷Location of search pointer at both start and finish of command execution.

## EDITOR LANGUAGE

The EDITOR language is composed of editing commands given by the user while working with a file and responses from the EDITOR subsystem to the user.

### Command Format

An EDITOR command may be a single verb only or a verb plus modifier. The modifiers specify variations from the standard operation of the verb and make up the "operand field" of the command. In the examples of command format below, everything following the verb is part of the operand field and, therefore, optional. When the operand field is used, the punctuation shown is required. No intervening blanks are permitted in the command format. (Capitalization of the verb is not required; it is done here to illustrate format.)

```
VERB
VERBm;r
VERBm:st
VERBm:st+st
VERBm:st-st
VERBm:st;r
VERBm:st,st
VERBm:st:st
VERBm:st,st;r
VERBm:st;r:st
```

Where:

m is the mode indicator or +/-
r is the repeat field
st is the string field

An abbreviated form of some verbs is allowed; the abbreviation is the initial letter of the verb.

```
BACKUP or B
COPY
CUT
DELETE or D
FIND or F
INSERT or I
PASTE
PRINT or P
REPLACE or R
CASE
```

The following verbs cannot have an operand field:

```
LINE or L      RUNOFF      STANDARD
STRING or S    VERIFY
BUILD          NOVERIFY
```

The use of the verbs and operands are fully explained and illustrated later in this section. The restrictions and usage rules that apply to the operand field are explained in "Operand Field of the Command" below.

The EDITOR responds to the commands with messages that inform the user when a command has been executed, a mistake in command format has been made, or the end of the file has been reached. These messages are described in "Responses from EDITOR", in this section.

## Operand Field of the Command

The operand field of the command can contain one or more of the following:

- Mode indicator (used only when a string field is used)
- Plus (+)n or minus (-)n to move line pointer forward or backward (line mode only) not applicable to BACKUP(B).
- String field, preceded by a colon
- Repeat field, preceded by a semicolon

If more than one of these items is used in a single command, the order must be as shown above.

Insertion or replacement text can also be a part of the operand. Refer to INSERT and REPLACE commands in this section.

The letter V appended to a command results in command verification. Refer to VERIFY and NOVERIFY commands in this section.

The letter B appended to the INSERT command permits insertions immediately before instead of after a specified line or string. Refer to the INSERT command in this section.

The following conventions concerning the effect of blanks and carriage returns are used by EDITOR in searching:

- Carriage returns can not be used for comparison purposes.
- Consecutive blanks in the file must be matched exactly by the blanks in the operand string field: _n_ consecutive blanks in the string field means _n_ consecutive blanks in the corresponding position of the character string in the file.

Certain commands permit two special forms of the string field to be used. String identifiers can be combined by the use of the Boolean AND or OR functions.

The searching conventions must be remembered when specifying string fields for searching to successfully locate the desired portion of the file.

## REPEAT FIELD

The repeat field specifies the number of times an operation is to be repeated. The field is always preceded by a semicolon and can contain a number or an asterisk. A number states the exact number of repetitions wanted; the asterisk (*) causes the operation to be repeated throughout the entire file. When the field is left blank, the operation is performed only once and the semicolon need not be used.

When the repeat field is used without a string field, the operation is always performed in the line mode.

The effect of the repeat field is explained in the detailed descriptions of each command (see "EDITOR Commands" in this section). However, a few brief examples are given below.

PRINT;5 (Prints five lines, beginning at the location of the search pointer.)

PRINT:/YOU/;3 (Prints the first three lines encountered beginning with the characters YOU. This would include YOUR, YOURS, YOU'RE, etc.)

PRINTS:/YOU/;3 (Prints the lines containing the first three occurrences of the characters YOU. This results in three lines of print, possibly the same line repeated if all three occurrences are in the same line.)

PRINT;* (Prints the complete file from the location of the search pointer.)

The difference between string and line mode requirements is as follows:

| LINE | STRING |
|---|---|
| The line field must contain the initial characters of the line as it appears in the file. | The string field can contain any characters in the line. (Caution: If the string to be operated upon appears twice in the same line and the second occurrence is where the change is to be made, be sure to include enough characters from the preceding or following word to make the string unique.) |

#NO MODE

The #NO mode allows a user to print a line numbered file without printing the line numbers. This mode is reset by typing #YES.

IGNORE MODE

The IGNORE mode allows the user to disregard line numbers while making modifications to a file using string functions. During execution, each line is scanned commencing with the first character of the line. The first nonnumeric character encountered is established as the first character of the line. To reset the mode to normal, the user types NOIG.

CAUTION: When the first character of the line is a numeric, some nonnumeric character should be inserted following the line number.

## Responses from EDITOR

- (hyphen)

The last command has been executed and EDITOR is ready to accept either another EDITOR command or a Time Sharing System command.

ENTER

This response to a REPLACE, INSERT, or BUILD command informs the user that the replacement, insertion, or additional text can now be entered.

An asterisk appears after the ENTER response, indicating that the time-sharing data collector now accepts text entry.

LIMIT REACHED

This message occurs only when a repeat field is used with an INSERT or REPLACE command and the text being entered exceeds the buffer capacity. All text input before the LIMIT REACHED message is entered into the file as many times as specified by the last repeat field. The search pointer will be at the last location altered.

To continue inserting or replacing text, back up and find the starting point, repeat the REPLACE or INSERT command and continue entering the rest of the text.

END OF FILE

This message occurs when the search pointer has reached the end of the file. This is the normal response to a command with an * in the repeat field. It also occurs when the specified string field does not appear in the file.

Following this message, a BACKUP command should be given if more work is to be done on the file.

COMMAND UNKNOWN

EDITOR does not recognize the command, either because it is illegal or because it is misspelled. This response may cause the EDITOR search pointer to be repositioned to the beginning of the current file. To return to the place in the file where the faulty command was given, the user can make use of the FIND command.

STRING ERROR

The command contains one or more of the following errors:

(1) The string mode has been specified but no string field has been entered.

(2) The ending delimiter is missing.

(3) A character(s) has been typed on the same line following the final delimiter.

REPEAT ERROR

The repeat field contains a character other than a number or *. Retype the command correctly.

END OF FILE - REQUEST EXECUTED XX TIMES

The above message occurs when a repeat field is used and the repeat field specified is greater than the number of occurrences in the file or the repeat field is an asterisk. XX represents the number of times the specified function was executed.

PASTE NOT EXECUTED, NO DATA

The above message occurs as a result of one of two reasons:

(1) Either the user failed to cut or copy data prior to issuing a PASTE command, or
(2) A system malfunction occurred preventing the data specified from being cut or copied.

TEXT INSERTION ERROR

This message occurs as a result of a missing delimiter or text following the terminating delimiter.

OPERAND ERROR

This message occurs as a result of an operand error. Either an inappropriate operand, an operand utilized where operands are not permitted, or an operand was expected and not found.

UNABLE TO CUT/COPY, NO FILE SPACE

Unable to cut or copy due to a lack of temporary file space to store the cut or copied data.

CUT/COPY TRUNCATED, PERFORM PASTE TO CONTINUE

The above message occurs as a result of an extensive amount of text being cut or copied, causing the cut/copy file to overflow. Performing a paste function following this message allows continued use of cut/copy file.

```
<50> WORK FILE, TABLE FULL, STATUS 36
<50> WORK FILE, SYSTEM LOADED, STATUS 40
<50> WORK FILE, STATUS 10
<52> CURRENT FILE NOT DEFINED
<50> NO FILE SPACE, STATUS 13
```

Refer to TSS General Information Manual, for message explanations.

## RUNOFF Format Control Words

RUNOFF format control words can be entered in the text file during the building or editing phase of the EDITOR subsystem to achieve such text format as spacing, indentation, and page numbering. These format control words remain imbedded within the text file but are removed in a printout of the file by way of the RUNOFF subsystem command REFORM. Refer to Section IV for descriptions of RUNOFF subsystem commands and format control words.

## TIME SHARING SYSTEM CONTROL COMMANDS

Time Sharing System control commands perform nonediting functions (e.g., saving or purging files, calling in other subsystems) for EDITOR. These control commands can be entered immediately after the appearance of the - response, the "-" indicating system readiness to accept a command. Time Sharing System control commands and their application to the EDITOR subsystem are described in the TSS General Information Manual.

## EDITOR COMMANDS

The EDITOR commands are described below in the following order (note the permissible abbreviations):

STRING
LINE
BACKUP or B
PRINT or P
FIND or F
REPLACE or R
DELETE or D
INSERT or I
COPY
CUT
PASTE
BUILD
RUNOFF
VERIFY
NOVERIFY
CASE
STANDARD
MODE or M
TRANSPARENT or T
MARK
AFTLIN or A
BEFLIN or BEFL

EDITOR commands can be employed singly or in multiples, the only restriction being that the one or more commands be contained on a single line. With use of the single command, a "-" response is issued upon command execution and control is returned to the user. With multiple command use, the series of commands are executed before the "-" response is issued and control returned to the user. Commands (and accompanying operands, if any) in a multiple command line must be separated by one or more blanks. For example,

```
-F   P;5   B;5   P;3   D;5
```

An unsuccessful command execution in a multiple command aborts the execution of any remaining commands.

NOTE: The slant is used as a delimiter to illustrate EDITOR commands below. Any keyboard character, except a blank or control character, can be used as a delimiter.

### STRING Command

The STRING command causes the commands which follow to be executed in the string mode. It is equivalent to adding the S mode indicator to each command typed.

NOTE: Since the first four characters of STRING and STRIP are equivalent, the system command STRIP does not function from within the Text Editor; i.e., the string mode is set instead.

STRING never takes an operand field; however, if the commands which follow STRING do not have a string field included, they operate as if in the line mode.

```
              -STRING
       ┌──────-PRINT;6
       │    <> A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
       │       TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
       │       TASK. EACH INSTRUCTION IS PERFORMED IN THE
       │       SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY,
       │       THE COMPUTER PROCESSES AND PRODUCES INFORMATION
       │     ▷ AS DIRECTED BY THE PROGRAM.
       │       -BACKUP
     ┌─┼──────-REPLACE:/TASK/
     │ │       ENTER
     │ │    <|> *JOB
     │ │       *(carriage return)
   ┌─┼─┼──────-PRINT:/JOB/
   │ │ │       JOB. EACH INSTRUCTION IS PERFORMED IN THE
   │ │ │       -
   │ │ │
   │ │ └───── Line mode action
   │ └─────── String mode action
   └───────── String mode action
```

## LINE Command

The LINE command counteracts the effect of STRING by placing EDITOR in the line mode, its normal mode of operation. All commands operate in line mode unless the S mode indicator is added to the verb. LINE never has an operand field and is only used to nullify the STRING command.

## BACKUP Command

The BACKUP command moves the search pointer backward the number of lines specified in the operand field. If the operand field is blank, the pointer backs up to the beginning of the file. The use of BACKUP is illustrated in the examples given for other EDITOR commands.

The formats and execution are as follows:

| Command | Execution |
|---|---|
| BACKUP | Backup search pointer to beginning of file. |
| BACKUP;n | Backup search pointer _n_ consecutive lines. |

## LIMIT Function

The LIMIT function allows the user to specify a portion of a line numbered file within which all further verb operations are restricted.

SAMPLE USAGE:

```
LIMIT:/203/,/506/ or L:/203/,/506/
```

This mode establishes a subset of a file wherein the line numbered 203 is the first line and line number 506 is the last line. All future function of verbs are executed only within the range specified, i.e., lines which begin with numbers between 203 and 506.

When specifying "LIMIT", if the current line pointer is located outside of the range specified, the pointer will be automatically positioned within the limits range. When returning to the normal "NORM" mode, the line pointer will remain pointing at the last line accessed while in the "LIMIT" mode.

It is possible to insert "BEFORE" or "FOLLOWING" the specified limited range. If the line numbers of the line(s) inserted are less than or greater than (respectively) the original limit range, the specified limits remain in effect. However, if the line numbers of the line(s) inserted are encompassed within the original limits range, the range is adjusted to include those lines inserted.

NOTE: The LIMIT mode can not function with Automatic Line Numbering (#AUTO) or RESEQUENCE.

To reset the mode to normal, the user need only type in NORM.

## WHERE Function

The WHERE function provides the user with the current internal block number and the location of the current line within the block. For example:

```
-WHERE          (User types "WHERE" - short form "W" is acceptable)

OCTL BLK#xxxx   (Text Editor identifies the current block
                number)
RCW=nnn         (Text Editor identifies the address of the
                current line)
```

Where: xxxx is the current block number (octal) and nnn is the address of the current line RCW (record control word) within the current block

Usage is principally technical, where a user desires to interrogate octal data within a file, or to patch data within a file. The octal block number cannot exceed 7777, otherwise the count will roll over, providing a false block number.

## Octal Function

The "OCTL d" function allows the user to designate a unique character (d) to precede an octal number. For example:

```
-OCTL $     (User identifies the dollar sign as the octal
            delimiter to be used)

-P          (Print the current line)

.......on at 9.084 - off at 9.140 on 06/24/75.............

-RS:/at/;2:/$100/       (Replace the string "at" twice with the
                         octal character 100 ( ))

-P          (Print the current line)

.......on  9.084 - off  9.140 on 06/24/75.............
```

The OCTL delimiter is functional within the build mode of the Text Editor providing the mode was set prior to entering BUILD. For example:

```
-OCTL %         (User defines the percent sign as the octal delimiter)

-BUILD          (User enters the BUILD mode)

ENTER           (Text Editor "ENTER" command)

*.......on %100 9.084 - off %100 9.140 on 06/24/75.............
*cr             (User exits the BUILD mode)

-FV;*           (Position to the last line of the file and print)

.......on @ 9.084 - off @ 9.140 on 06/24/75.............

END OF FILE - REQUEST EXECUTED   1 TIMES
```

Caution: No tests are made to determine the validity of the octal character.

NOCT nullifies the octal function.

## Column Function

The function COLS:(xxx-yyy) allows the user to restrict string searches and modifications in a horizontal direction; i.e., to a specific range of character positions within one or all lines (depending on commands used). It is particularly useful if data is in columnar (tabular) format. For example:

```
-COLS:(9-11)   (User restricts the horizontal range to the
                characters located within columns 9 through
                11 (3 characters) inclusive.)

-P;2           (Print two lines)

1234567890123456789012345678 90.........
abc def abc def abc def abc.............

-RS:/abc/:/xyz/        (Replace the string abc with xyz)

-P             (Print the current line)

abc def xyz def abc def abc.............
```

NOTE: The string "abc" in columns 1, 2, and 3 was not affected since column function was restricted to columns 9, 10 and 11. A repeat factor is acceptable.

O: MAy NOT use another editor command on same line after COLS

Limitations:
1. Character string must start in the first column specified and terminate in the last column specified.
2. Multiple occurrences of strings within column parameters are not permitted.
3. Multiple column parameters are not permitted.
4. Only numerics are permitted within parentheses; use of characters other than numerics result in error messages.

NOCO nullifies the column function.

## MASK Function

The MASK function allows the user to manipulate a string without disturbing the surrounding characters. For example:

```
-MASK #         (User sets the "MASK" mode using the number
                sign as the delimiter)

-P;2            (Print two lines)

..........DATANET355.................
.................DATANET305................

-B;1            (Back the line pointer up one line)


IVS;/NET###/;2:/?/         (Insert and verify a question mark
                           following the string "NET" followed
                           by any three characters, do it
                           twice)

..........DATANET355?...................

.................DATANET305?................
```

Limitation: The mask character is only acceptable in the field containing the string to be worked on. It is not acceptable in the replacement field as a "mask" character.

NOMA nullifies the mask function.

## Occurrence Function

The use of the "O" operand allows the user to operate on a specific occurrence of a string. The use of the additional repeat field (;n) specifies which occurrence. For example:

Suppose a line contained the following repetitive data:

```
D.....D.....D.....D.....D.....D.....D.....D.....D.....D.....
```

In the above example it would be extremely difficult (if not impossible) to access the sixth occurrence of the string "D" without replacing the entire line.

With the use of the "Occurrence" modifier, replacement of the character would be performed as follows:

```
-RVO:/D/;6:/X/      (Replace and verify the sixth occur-
                    rence of the character "D" with "X")

D.....D.....D.....D.....D.....X.....D.....D.....D.....D.....
```

## PRINT Command

The PRINT command is used when either a selected portion of a file or the entire file is to be printed. The user can vary the PRINT command to print any one of the following:

- The entire file
- Any number of consecutive lines
- Any number of lines containing a given character string or strings
- From one point to another
- A single line

The formats and execution are as follows:

| Command | Execution |
|---|---|
| PRINT | Print one line. |
| PRINT;n | Print n consecutive lines. |
| PRINT-j;n | Backup the line pointer j lines and print n lines. |
| PRINT+j;n | Move the line pointer forward j lines and print n lines. |
| PRINT;* | Print entire file. |
| PRINT:/xxx/ | Print line identified by xxx. |
| PRINT:/xxx/;n | Print the next n lines identified by xxx. (* can be used instead of n to print all such lines.) |
| PRINT:/xxx/,/yyy/ | Print the block of lines starting with the line identified by xxx through the line identified by yyy. (A repeat field can be used to print n or all such lines.) |
| PRINTS:/yyy/ | Print line containing specified string. |
| PRINTS:/yyy/;n | Print n lines containing the specified string. (* can be used to print all lines containing the specified string.) |

| | |
|---|---|
| PRINTS:/yyy/,/zzz/ | Print from line containing string yyy to line containing string zzz, inclusive. (A repeat field can be used with this form also.) |
| PRINT:/xxx/+/yyy/+... | Print line containing all of the specified (a maximum of five) strings. (A repeat field can be used to print n or all such lines.) |
| PRINT:/xxx/-/yyy/-... | Print line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to print n or all such lines.) |

To print the complete file, use the PRINT command in line mode with the asterisk in the repeat field. Printing begins at the location of the search pointer and continues to the end of the file.

```
   -PRINT;*
 ◇PROGRAMMING LANGUAGES

  HUMAN LANGUAGES ARE IMPRACTICAL FOR PREPARING
  COMPUTER PROGRAMS BECAUSE THESE LANGUAGES
  CONTAIN MANY AMBIGUITIES AND REDUNDANCIES;
  THE COMPUTER INTERPRETS LANGUAGE ABSOLUTELY
  LITERALLY. BY THE SAME TOKEN, MACHINE
  LANGUAGES ARE ALSO IMPRACTICAL BECAUSE THEY ARE
  DIFFICULT FOR PEOPLE TO USE. MOST PROGRAMMING
  LANGUAGES ARE COMPROMISES BETWEEN HUMAN AND
  MACHINE LANGUAGES.

 ▷END OF FILE
```

To print a single line, use the PRINT command in line mode, with or without a string field. If no string field is specified, the line where the search pointer is located is printed.

```
-PRINT
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
-
```

When a string field is specified, the line identified by the string is printed. The string field must contain characters unique to the beginning of the line and only one string field can be used.

```
-BACKUP
-PRINT:/HUMAN/
HUMAN LANGUAGES ARE IMPRACTICAL FOR PREPARING
-
```

To print any number of consecutive lines, use PRINT in the line mode with a repeat field. Printing begins at the location of the search pointer.

```
     -BACKUP
     -PRINT;3
    ◇COMPUTER PROGRAMS

 ┌  ▷A COMPUTER PROGRAM IS A SET OF INSTRUCTION THAT
 │   -
 │
 └───Line space inserted during build
```

To print a specified string, use PRINTS with a string field and with or without a repeat field.

```
     -PRINTS:/SHAR/
     TIME-SHARING SYSTEM
     -PRINTS:/SHAR/;4
     TIME-SHARING SYSTEM
     THE TIME-SHARING SYSTEM USES A TECHNIQUE BY
     THUS, TIME-SHARING PERMITS A USER TO WORK
     MANY OTHERS AT THE SAME TIME SHARE THIS
     -
```

To print from point-to-point, use PRINTS and two string fields.

```
     -PRINTS:/TIME/,/USE./
    ◇TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
     COMPUTER AND USER, PERMITTING THE DIALOGUE
     TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
     THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
     DATA IS FED FROM THE TERMINAL DIRECTLY TO
     THE COMPUTER AND ANSWERS ARE RECEIVED
     QUICKLY AT THE SAME TERMINAL.

     IF THE PROGRAM CONTAINS A MISTAKE, THE
     COMPUTER INFORMS THE USER.

     THE PROGRAM CAN BE CORRECTED OR CHANGED BY
     THE USER AS IF HE WERE CONVERSING BY PHONE,
     EXCEPT IN THIS CASE, THE CONVERSATION IS
     TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
    ▷OF TERMINAL IN USE.
     -
```

The first string field must contain data unique to the first line printed and the second string field must be unique to the last line printed. In the above example, if the second string field did not contain the period after USE, only the two lines of text, through the line containing the word USER, would have been printed.

Two special forms of the operand are permissible with the PRINT command to identify lines containing specified strings. These forms of the operand are referred to as Boolean AND and OR functions. The operand can consist of up to five separate strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the print. For example, with d representing a delimiter the format is

```
PRINT:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the print. For example, with d representing a delimiter, the format is

```
PRINT:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

```
-PRINT;*
A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK. EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY,
THE COMPUTER PROCESSES AND PRODUCES INFORMATION
AS DIRECTED BY THE PROGRAM.

A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.

     THE PROGRAM MUST BE SUBMITTED TO THE
     COMPUTER IN A LANGUAGE THAT THE
     COMPUTER RECOGNIZES.

     ALL LANGUAGE INSTRUCTIONS MUST BE
     COMPLETE AND BE PRECISELY STATED.

END OF FILE
BACKUP


-PRINT:/COMPUTER/+/PRODUCES/
THE COMPUTER PROCESSES AND PRODUCES INFORMATION
-BACKUP
-FIND:/TWO/-/PRIMARY/-/REQUIREMENTS/
-PRINT
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
```

## FIND Command

The FIND command moves the search pointer through the file. FIND may be used with or without an operand field.

If in doubt as to where the search pointer is located, give the PRINT command with no operand field. The resulting printout is the line pointed to by the search pointer. It is advisable, when editing a file in which the specified string may appear more than once, to print the line before changing the file, in order to ensure that the change is made in the right place.

The repeat field can be used with a string field in the FIND command. The search and comparison continues until the comparison is made as many times as indicated. When execution is completed, the "-" response appears. If the repeat field is used without a string field, the search pointer moves forward n number of lines as indicated by the repeat field.

The formats and execution are as follows:

| Command | Execution |
|---|---|
| FIND | Advance search pointer one line. |
| FIND;n | Advance search pointer n lines. |
| FIND:/xxx/ | Find line identified by xxx. |
| FIND:/xxx/;n | Find nth line identified by xxx. |
| FINDS:/yyy/ | Find line containing specified string. |
| FINDS:/yyy/;n | Find the line containing the nth occurrence of the specified string. |
| FIND:/xxx/+/yyy/+... | Find line containing all of the specified strings. (A repeat field can be used to find n or all such lines.) |
| FIND:/xxx/-/yyy/-... | Find line containing one of the specified strings. (A repeat field can be used to find n or all such lines.) |

To find a specified string, not at the beginning of the line, use FIND in the string mode.

```
-FINDS:/MUST/
-PRINT
     THE PROGRAM MUST BE SUBMITTED TO THE
-BACKUP;4
-PRINT;*
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.

     THE PROGRAM MUST BE SUBMITTED TO THE
     COMPUTER IN A LANGUAGE THAT THE
     COMPUTER RECOGNIZES.

END OF FILE
```

To find a string past the point where it next occurs, use FIND in the string mode with a repeat field.

```
-PRINT;6
COMPUTER PROGRAMS

A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK. EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY
-B
-FINDS:/IS/;3
-PRINT
TASK. EACH INSTRUCTION IS PERFORMED IN THE
```

To find a specified number of lines, use FIND in line mode with a repeat field. The number in the repeat field includes the line at which the search pointer is located at the beginning of execution (unless FIND is used without a string field, in which case line 1 is the line following).

```
-PRINT;4
THE TIME-SHARING SYSTEM USES A TECHNIQUE BY
WHICH PROGRAMS ARE HANDLED IN PARALLEL. A
SUPERVISORY PROGRAM ACTS AS A CONTROLLER OF
THESE PROGRAMS, CONTROLLING "STOP" AND "GO"
-FIND;1
-PRINT
SIGNALS TO INPUTS FROM TERMINALS AND
```

Two special forms of the operand are permissible with the FIND command to identify lines containing specified strings. These forms of the command are referred to as the Boolean AND and OR functions. The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line can be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the find. For example, with d representing a delimiter, the format is

```
FIND:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the find. For example, with d representing a delimiter, the format is

```
FIND:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

Examples of the use of these forms of the operand are given with the description of the PRINT command above.

REPLACE Command

The REPLACE command allows the user to replace any number of characters, words, or lines of text with new text of any length. REPLACE may or may not have an operand field. If no operand field is given, the line where the search pointer is located is replaced.

The operand field can take one of two forms, depending on the length of the replacement text. The first four formats illustrate the format to be used when the operand field cannot be contained in one line. The remaining three formats illustrate the use of REPLACE with short strings.

The formats and execution are as follows:

| Command | Execution |
|---|---|
| REPLACE | Replace line at which search pointer is currently located. (A repeat field can be used with this form.) |
| REPLACE:/xxx/ | Replace line identified by xxx. |
| REPLACE:/xxx/;n | Replace the next n lines identified by xxx. (* can be used instead of n to replace all such lines.) |
| REPLACE:/xxx/,/yyy/ | Replace the block of lines starting with the line identified by xxx through the line identified by yyy. |
| REPLACES:/yyy/ | Replace specified string. |
| REPLACES:/yyy/;n | Replace n successive occurrences of the specified string. (* can be used instead of n to replace all such occurrences.) |
| REPLACES:/yyy/,/zzz/ | Replace text between points yyy and zzz, inclusive. (A repeat field can be used with this form also.) |
| REPLACE:/stl/+.../stn/ | Replace line containing all of the specified (a maximum of five) strings. (A repeat field can be used to replace n or all such lines.) |
| REPLACE:/stl/-.../stn/ | Replace line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to replace n or all such lines.) |

Following the REPLACE commands above, the system responds with ENTER. The replacement text is then typed in. Following the ENTER response, the replacement text must include all desired blanks and carriage returns. Replacement text is typed on lines following ENTER. When text entry is complete, a carriage return in response to the asterisk generates the - response.

In string mode, the carriage return on the last line of text is ignored. When replacing short strings of text, the formats shown below can be used.

NOTE: The command and the entire operand field must be on the same line. This format does not accept a carriage return before the final delimiter. The ENTER response is not given with this use of the command.

| Command | Execution |
|---|---|
| REPLACE:/xxx/:/bbb/ | Replace line identified by xxx with the string (line) bbb. |
| REPLACE:/xxx/;n:/bbb/<br>R:/xxx/;/bbb | Replace the next n lines identified by xxx with the string (line) bbb. (* can be used instead of n to replace all such lines.) |
| REPLACES:/yyy/;n:/bbb/ | Replace n successive occurrences of the string yyy with string bbb. (* can be used instead of n to replace all such occurrences.) |
| REPLACES:/yyy/,/zzz/:/bbb/ | Replace text between points yyy and zzz, inclusive, with string bbb. (A repeat field can be used with this form also.) |

To replace a string of characters, use REPLACE in the string mode with a string field and with or without a repeat field. Replacement begins at the first character position specified in the operand string field. If a repeat field is specified, n identical replacements are performed (unless end-of-file is encountered first).

```
-PRINT
A PROGRAM MUST MEET TWO PRIMERY REQUIREMENTS
-REPLACES:/ERY/:/ARY/
-PRINT
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
-
```

To replace a complete line, use REPLACE in the line mode, with or without a string field and/or repeat field. The string field, when used, must contain the characters unique to the beginning of the line. When no string or repeat field is given, the line where the search pointer is located is replaced.

Example 1

```
-PRINT
TIME-SHARING LANGUAGES
-REPLACE:/TI/:/TIME-SHARING SYSTEM/
-PRINT
TIME-SHARING SYSTEM
-
```

Example 2

```
-PRINT
TIME-SHARING LANGUAGES
-REPLACE
ENTER
*TIME-SHARING SYSTEM
*(carriage return)
-PRINT
TIME-SHARING SYSTEM
-
```

When the repeat field is used, the lines beginning with the specified string are replaced the indicated number of times. If no string field is given, the indicated number of lines is replaced.

```
-PRINT;14
THE TIME-SHARING SYSTEM USES A TECHNIQUE BY
WHICH PROGRAMS ARE HANDLED IN PARALLEL.  A
SUPERVISORY PROGRAM ACTS AS A CONTROLLER OF
THESE PROGRAMS, CONTROLLING "STOP" AND "GO"
SIGNALS TO INPUTS FROM TERMINALS AND
PREVENTING DEMANDS OF ONE TERMINAL FROM
INTERFERING WITH DEMANDS OF OTHER TERMINALS.
THUS, TIME-SHARING PERMITS A USER TO WORK
DIRECTLY WITH THE COMPUTER, WHETHER IT IS
WITHIN HIS SIGHT OR THOUSANDS OF MILES
AWAY.  THE USER BELIEVES THAT HE HAS
EXCLUSIVE USE OF THE COMPUTER, EVEN THOUGH
MANY OTHERS AT THE SAME TIME SHARE THIS
ILLUSION.
-BACKUP;13
-REPLACE;2
ENTER
*A TIME-SHARING SYSTEM
*(carriage return)
-PRINT;5
A TIME-SHARING SYSTEM
SUPERVISORY PROGRAM ACTS AS A CONTROLLER OF
THESE PROGRAMS, CONTROLLING "STOP" AND "GO"
SIGNALS TO INPUTS FROM TERMINALS AND
PREVENTING DEMANDS OF ONE TERMINAL FROM
```

To replace from point-to-point, use REPLACE in the string mode with two string fields. A repeat field can be used if desired.

```
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER, PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE  SAME TERMINAL.
```

```
THE  PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE OF
TERMINAL IN USE.

END OF FILE
B;13
-REPLACES:/SAME/,/THE/
ENTER
*SAME TERMINAL.
*(blank,carriage return)
*IF THE PROGRAM CONTAINS A MISTAKE, THE
*COMPUTER INFORMS THE USER.
*(blank,carriage return)
*THE
*(carriage return)
-B;11
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE  SAME TERMINAL.

IF THE PROGRAM CONTAINS A MISTAKE, THE
COMPUTER INFORMS THE USER.

THE  PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.

END OF FILE
```

Two special forms of the operand are permissible with the REPLACE command to identify lines containing specified strings. These forms of the command are referred to as the Boolean AND and OR functions. The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line can be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the replace. For example, with d representing a delimiter, the format is

```
REPLACE:dSTRING1d+...+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the replace. For example, with d representing a delimiter, the format is

```
REPLACE:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## DELETE Command

The DELETE command allows the user to delete any number of characters, words, or lines from his file. The operand field of the command specifies the text to be deleted. If no operand field is given, the line where the search pointer is located is deleted.

The formats and execution are as follows:

| Command | Execution |
|---|---|
| DELETE or DELETE;n | Delete line or lines at which search pointer is currently located. |
| DELETE:/xxx/ | Delete line identified by xxx. |
| DELETE:/xxx/;n | Delete the next n lines identified by xxx. (* can be used instead of n to delete all such lines.) |
| DELETE:/xxx/,/yyy/ | Delete the block of lines starting with the line identified by xxx through the line identified by yyy. (A repeat field can be used to delete n or all such lines.) |
| DELETES:/yyy/ | Delete specified string. |
| DELETES:/yyy/;n | Delete n occurrences of specified string. (* can be used instead of n to delete all such occurrences.) |
| DELETES:/yyy/,/zzz/ | Delete text between points yyy and zzz, inclusive. (A repeat field can be used with this form also.) |
| DELETE:/stl/+.../stn/ | Delete line containing all of the specified (a maximum of five) strings. (A repeat field can be used to delete n or all such lines.) |
| DELETE:/stl/-.../stn/ | Delete line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to delete n or all such lines.) |

To delete a string of characters, use DELETE in the string mode with or without a repeat field.

```
-PRINT
(HAVE ALL INSTRUCTIONS EXECUTED 0) ON A COMPUTER.
-DELETES:/ 0/
-PRINT
(HAVE ALL INSTRUCTIONS EXECUTED) ON A COMPUTER.
-
```

To delete from point-to-point, use delete in the string mode with two string fields and with or without a repeat field. All data between and including the two points indicated is deleted.

```
-PRINT;4
COMPUTER PROGRAMS BECAUSE THESE LANGUAGES
CONTAIN MANY AMBIGUITIES AND REDUNDANCIES;
THE COMPUTER INTERPRETS LANGUAGE ABSOLUTELY
LITERALLY.  BY THE SAME TOKEN, MACHINE
-B
-DS:/THE C/,/.  /
-B
-P;4
COMPUTER PROGRAMS BECAUSE THESE LANGUAGES
CONTAIN MANY AMBIGUITIES AND REDUNDANCIES;
BY THE SAME TOKEN, MACHINE
LANGUAGES ARE ALSO IMPRACTICAL BECAUSE THEY ARE
-
```

To delete one or more lines, use DELETE in line mode, with or without a string field and/or repeat field. If both a string field and repeat field are used, the indicated number of lines beginning with the specified string are deleted. If no string field is used with the repeat field, the indicated number of lines is deleted, beginning at the location of the search pointer.

```
-PRINT;4
HUMAN LANGUAGES ARE IMPRACTICAL FOR PREPARING
COMPUTER PROGRAMS BECAUSE THESE LANGUAGES
CONTAIN MANY AMBIGUITIES AND REDUNDANCIES;
THE COMPUTER INTERPRETS LANGUAGE ABSOLUTELY
-B;3
-D;3
-PRINT;2
THE COMPUTER INTERPRETS LANGUAGE ABSOLUTELY
LITERALLY.  BY THE SAME TOKEN, MACHINE
-
```

To delete all lines having a common beginning, use DELETE in line mode with a repeat field. Note that in the following example the sentence "ALL LANGUAGE INSTRUCTION MUST BE" is not deleted because the letter A is preceded by blanks.

```
-PRINT;*

COMPUTER PROGRAMS

A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK.  EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM.  IN THIS WAY,
THE COMPUTER PROCESSES AND PRODUCES INFORMATION
AS DIRECTED BY THE PROGRAM.

A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.

     THE PROGRAM MUST BE SUBMITTED TO THE
     COMPUTER IN A LANGUAGE THAT THE
     COMPUTER RECOGNIZES.

     ALL LANGUAGE INSTRUCTION MUST BE
     COMPLETE AND BE PRECISELY STATED.
```

```
END OF FILE
B
-DELETE:/A/;*

END OF FILE - REQUEST EXECUTED   3 TIMES
B
-PRINT;*

COMPUTER PROGRAMS

TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK.  EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM.  IN THIS WAY,
THE COMPUTER PROCESSES AND PRODUCES INFORMATION

BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.

     THE PROGRAM MUST BE SUBMITTED TO THE
     COMPUTER IN A LANGUAGE THAT THE
     COMPUTER RECOGNIZES.

     ALL LANGUAGE INSTRUCTION MUST BE
     COMPLETE AND BE PRECISELY STATED.

END OF FILE
```

Two special forms of the operand are permissible with the DELETE command to identify lines containing specified strings. These forms of the command are referred to as "ANDing" and "ORing". The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the delete. For example, with d representing a delimiter, the format is

```
DELETE:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the delete. For example, with d representing a delimiter, the format is

```
DELETE:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## INSERT Command

The INSERT command allows the user to insert any number of characters, words, or lines into his file. The operand field of the INSERT command specifies the point after which the insertion is to be made and can take one of two forms, depending on the length of the text being inserted.

The first list below illustrates the format to be used when the operand field cannot be contained on one line. The system responds to the INSERT command with the word ENTER. The text to be inserted is then typed on lines following ENTER. When text entry is complete, a carriage return following the asterisk generates the "-" response. The second list illustrates the use of INSERT with short strings; the ENTER response is not given in this use of the command.

The formats and execution are as follows:

| Command | Execution |
|---|---|
| INSERT | Insert after the line at which the search pointer is currently located. |
| INSERT:/xxx/ | Insert after the line identified by xxx. |
| INSERT:/xxx/;n | Insert after each of the next n lines identified by xxx. (* can be used instead of n to insert after all such lines.) |
| INSERTS:/yyy/ | Insert after point yyy. |
| INSERTS:/yyy/;n | Insert after each of n successive occurrences of point yyy. (* can be used instead of n to insert after all such occurrences.) |
| INSERTS:/yyy/,/zzz/ | Insert after point zzz. (A repeat field can be used with this form.) |
| INSERT:/stl/+.../stn/ | Insert after line containing all of the specified (a maximum of five) strings. (A repeat field can be used to insert after n or all such lines.) |
| INSERT:/stl/-.../stn/ | Insert after line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to insert after n or all such lines.) |

When inserting short strings of text, the following formats can be used.

NOTE: The command and the entire operand field must be on the same line. This format does not accept a carriage return before the final delimiter.

| Command | Execution |
|---|---|
| INSERT:/xxx/:/bbb/ | Insert string bbb after the line identified by xxx. |
| INSERT:/xxx/;n:/bbb/ | Insert string bbb after each of the next n lines identified by xxx. (* can be used instead of n to insert after all such lines.) |
| INSERTS:/yyy/:/bbb/ | Insert string bbb after point yyy. |
| INSERTS:/yyy/;n:/bbb/ | Insert string bbb after each of n successive occurrences of point yyy. (* can be used instead of n to insert after all such occurrences.) |
| INSERTS:/yyy/,/zzz/:/bbb/ | Insert string bbb following point zzz. (A repeat field can be used with this form.) |

To insert one or more lines, use INSERT in the line mode with or without a string field and/or repeat field. If no string field is used, the insertion is made after the line where the search pointer is located. For insertions of more than one line, each new line must be followed by a carriage return to prevent it from running into the next line.

```
-PRINT;6
QUICKLY AT THE SAME TERMINAL.
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.
-B;5
-INSERT
ENTER
*IF THE PROGRAM CONTAINS A MISTAKE, THE
*COMPUTER INFORMS THE USER.
*(carriage return)
-B;3
-PRINT;*
QUICKLY AT THE SAME TERMINAL.
IF THE PROGRAM CONTAINS A MISTAKE, THE
COMPUTER INFORMS THE USER.
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.

END OF FILE
```

To insert a string of characters, use INSERT in the string mode with a string field and with or without a repeat field. The string field must identify the point after which the insertion is to be made.

```
-PRINT;4
THE TIME-SHARING SYSTEM USES A TECHNIQUE BY
WHICH PROGRAMS ARE HANDLED IN PARALLEL.
THUS, TIME-SHARING PERMITS A USER TO WORK
DIRECTLY WITH THE COMPUTER, WHETHER IT IS
-B
```

```
-INSERTS:/LEL./
ENTER
* A
*SUPERVISORY PROGRAM ACTS AS A CONTROLLER OF
*THESE PROGRAMS, CONTROLLING "STOP" AND "GO"
*SIGNALS TO INPUTS FROM TERMINALS AND
*PREVENTING DEMANDS OF ONE TERMINAL FROM
*INTERFERING WITH DEMANDS OF OTHER TERMINALS.
*(carriage return)
-B
-PRINT;9
THE TIME-SHARING SYSTEM USES A TECHNIQUE BY
WHICH PROGRAMS ARE HANDLED IN PARALLEL.  A
SUPERVISORY PROGRAM ACTS AS A CONTROLLER OF
THESE PROGRAMS, CONTROLLING "STOP" AND "GO"
SIGNALS TO INPUTS FROM TERMINALS AND
PREVENTING DEMANDS OF ONE TERMINAL FROM
INTERFERING WITH DEMANDS OF OTHER TERMINALS.
THUS TIME-SHARING PERMITS A USER TO WORK
DIRECTLY WITH THE COMPUTER, WHETHER IT IS
-F:/THE PROGRAM/
-PRINT
THE PROGRAM BE CORRECTED OR CHANGED BY
-INSERTS:/RAM /:/CAN /
-P
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
```

To insert at the beginning of the file, use INSERTB in the line mode with no operand field.

```
-PRINT;3
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.
-B
-INSERTB
ENTER
*COMPUTER PROGRAMS
*(blank,carriage return)
*A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
*TELLS A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
*TASK. EACH INSTRUCTION IS PERFORMED IN THE
*SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY,
*THE COMPUTER PROCESSES AND PRODUCES INFORMATION
*AS DIRECTED BY THE PROGRAM.
*(carriage return)
-B
-PRINT;11
```

```
COMPUTER PROGRAMS

A COMPUTER PROGRAM IS A SET OF INSTRUCTIONS THAT
TELL A COMPUTER HOW TO ACCOMPLISH A SPECIFIC
TASK.  EACH INSTRUCTION IS PERFORMED IN THE
SEQUENCE SPECIFIED BY THE PROGRAM. IN THIS WAY,
THE COMPUTER PROCESSES AND PRODUCES INFORMATION
AS DIRECTED BY THE PROGRAM.
A PROGRAM MUST MEET TWO PRIMARY REQUIREMENTS
BEFORE IT CAN BE RUN (HAVE ALL INSTRUCTIONS
EXECUTED) ON A COMPUTER.
```

The INSERT command, in conjunction with the #TAPE command, allows the user to insert text from paper tape in the file at any point in the file. At the selected point (as determined by the operand of the INSERT command), the user activates the paper tape reader to read in the tape after the appearance of the ENTER response. Upon termination of tape read, the user gives a carriage return in response to the asterisk and the - response appears.

```
-INSERT (appropriate operand)
ENTER
*#TAPE
READY

(user activates paper tape reader and
text is read in from tape.)

*(carriage return)
-
```

Text may be alternatively inserted from the keyboard and from paper tape.

```
-INSERT (appropriate operand)
ENTER
*Text entered by user
*more text
*last line of text
*#TAPE
READY
(user activates paper tape reader and text
is read in from tape.)
*Text entered by user
*more text
*last line of text
*(carriage return)
-
```

The INSERT command, as indicated in the descriptions of the command above, provides for insertion of data following the specified line or string. An optional operand, the letter B, can be used with the INSERT command to achieve insertion before the specified line or string.

```
-STRING
-F
-P
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
-INSERTB:/THE/:/THEREFORE, /
-P
THEREFORE, THE PROGRAM  CAN BE CORRECTED OR CHANGED BY
-
```

Two special forms of the operand are permissible with the INSERT command to identify lines containing specified strings. These forms of the operand are referred to as the Boolean AND and OR functions. The operand can consist of up to five separate strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the insert. For example, with d representing a delimiter the format is

```
INSERT:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the insert. For example, with d representing a delimiter, the format is

```
INSERT:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## COPY Command

The COPY command allows the user to copy a specified portion of text and hold it in reserve for a PASTE command. The copied text is not removed from the file but is repeated at the location specified by PASTE. Several sequential COPY commands can be given and the collected text inserted with a single PASTE command. Examples of the use of COPY are included with the PASTE examples.

The format and execution are as follows:

| Command | Execution |
|---|---|
| COPY | Copy the line at which the search pointer is currently located. (A repeat field can be used with this form.) |
| COPY:/xxx/ | Copy line identified by xxx. |
| COPY:/xxx/;n | Copy the next n lines identified by xxx. (* can be used to copy all such lines.) |
| COPY:/xxx/,/yyy/ | Copy the block of lines starting with the line identified by xxx through the line identified by yyy. (A repeat field can be used to copy n or all such blocks of lines.) |
| COPYS:/yyy/ | Copy the line that contains the specified string. |
| COPYS:/yyy/;n | Copy n occurrences of the line that contains the specified string. (* can be used to copy all such lines.) |
| COPYS:/yyy/,/zzz/ | Copy text between points yyy and zzz, inclusive. (A repeat field can be used with this form also.) |
| COPY:/st1/+.../stn/ | Copy line containing all of the specified (a maximum of five) strings. (A repeat field can be used to copy n or all such lines.) |

| | |
|---|---|
| COPY:/stl/-.../stn/ | Copy line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to copy n or all such lines.) |

Two special forms of the operand are permissible with the COPY command to identify lines containing specified strings. These forms of the command are referred to as the Boolean AND and OR functions. The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the copy. For example, with d representing a delimiter, the format is

```
COPY:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the copy. For example, with d representing a delimiter, the format is

```
COPY:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## CUT Command

The CUT command performs the same functions as COPY, except the copied text is deleted from its present location. Examples of this are included with the PASTE examples. The formats and execution are as follows.

| Command | Execution |
|---|---|
| CUT | Copy and remove the line at which the search pointer is located. (A repeat field can be used with this form.) |
| CUT:/xxx/ | Copy and remove the line identified by xxx. |
| CUT:/xxx/;n | Copy and remove the next n lines identified by xxx. (* can be used to copy and remove all such lines.) |
| CUT:/xxx/,/yyy/ | Copy and remove the block of lines starting with the line identified by xxx through the line identified by yyy. (A repeat field can be used to copy and remove n or all such blocks of lines.) |
| CUTS:/yyy/ | Copy and remove the line that contains the specified string. |
| CUTS:/yyy/;n | Copy and remove n occurrences of the line that contains the specified string. (* can be used to copy and remove all such lines.) |
| CUTS:/yyy/,/zzz/ | Copy and remove text between points yyy and zzz, inclusive. (A repeat field can be used to copy and remove n or all such occurrences of text.) |

| | |
|---|---|
| CUT:/stl/+.../stn/ | Copy and remove line containing all of the specified (a maximum of five) strings. (A repeat field can be used to copy and remove n or all such lines.) |
| CUT:/stl/-.../stn/ | Copy and remove line or lines containing any one of the specified (a maximum of five) strings. (A repeat field can be used to copy and remove n or all such lines.) |

Two special forms of the operand are permissible with the CUT command to identify lines containing specified strings. These forms of the command are referred to as the Boolean AND and OR functions. The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the cut. For example, with d representing a delimiter, the format is

```
CUT:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the cut. For example, with d representing a delimiter, the format is

```
CUT:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## PASTE Command

The PASTE command inserts the collected CUT or COPY text into the specified location. In order to PASTE the copied text in more than one location, successive PASTE instructions must be used. Once a PASTE command has been executed, the next COPY or CUT command wipes out the previously accumulated COPY or CUT text.

| Command | Execution |
|---|---|
| PASTE | Insert text after the line at which the search pointer is currently located. |
| PASTE:/xxx/ | Insert text after the line identified by xxx. |
| PASTE:/xxx/;n | Insert text after each of the next n lines identified by xxx. (* can be used to insert after all such lines.) |
| PASTES:/yyy/ | Insert text after point yyy. |
| PASTES:/yyy/;n | Insert text after each of n successive occurrences of point yyy. (* can be used to insert after all such occurrences.) |

| | |
|---|---|
| PASTE:/stl/+.../stn/ | Insert text after all specified (a maximum of five) strings. (A repeat field can be used to insert text after line containing _n_ or all such lines.) |
| PASTE:/stl/-.../stn/ | Insert text after line containing any one of the specified (a maximum of five) strings. (A repeat field can be used to insert text after _n_ or all such lines.) |

To cut and paste one or more lines, use CUT in the line mode, with or without a string field and/or repeat field. If both a string field and repeat field are used, the indicated number of lines beginning with the specified string is copied, removed, and then inserted by PASTE. If no string field is used with the repeat field, the indicated number of lines is copied and removed, beginning at the location of the search pointer.

```
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER, PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE SAME TERMINAL.

IF THE PROGRAM CONTAINS A MISTAKE, THE
COMPUTER INFORMS THE USER.
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.

END OF FILE
B
-FIND:/QUICKLY/
-FIND;1
-CUT;3
-PASTE:/OF/
-B
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER, PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE SAME TERMINAL.
```

```
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.

IF THE PROGRAM CONTAINS A MISTAKE, THE
COMPUTER INFORMS THE USER.

END OF FILE.
```

To paste the same text in several locations, use CUT or COPY, then successive PASTE commands, one for each insertion needed. The example illustrates a form letter and mailing list contained in the same file. In this case, a continuous PASTE command is used, since each insertion is made following a line beginning with the same word.

```
-PRINT;*

WE TAKE GREAT PLEASURE IN ANNOUNCING
                 .
                 .
                 .
                 .
YOURS VERY TRULY,

COMPANY NAME
ADDRESS
CITY,STATE

MR. A. A. ADAMS
ADDRESS
CITY,STATE

DEAR MR. ADAMS:
                 .
                 .
                 .
MR. X. Y. ZILCH
ADDRESS
CITY,STATE

DEAR MR. ZILCH:

END OF FILE
B
-COPY:/ /,/CITY/
```

(The space character between the first set of delimiters causes the blank line at the beginning of the file to be included with the copied text.)

```
-PASTE:/DEAR/;*

END OF FILE - REQUEST EXECUTED   2 TIMES
B
-FIND:/MR./
-PRINT;*
MR. A. A. ADAMS
ADDRESS
CITY,STATE

DEAR MR. ADAMS:

WE TAKE GREAT PLEASURE IN ANNOUNCING
                     .
                     .
                     .
                     .
YOURS VERY TRULY,

COMPANY NAME
ADDRESS
CITY,STATE

                     .
                     .
                     .
DEAR MR. ZILCH:

WE TAKE GREAT PLEASURE IN ANNOUNCING
                     .
                     .
                     .
                     .
YOURS VERY TRULY,

COMPANY NAME
ADDRESS
CITY,STATE

END OF FILE
```

Two special forms of the operand are permissible with the PASTE command to identify lines containing specified strings. These forms of the command are referred to as the Boolean AND and OR functions. The operand can consist of up to five strings connected by plus signs for the AND form and minus signs for the OR form. The strings can be in any order; i.e., the fifth string in order of appearance in the line may be listed first in the operand.

For the AND form, the user lists strings and plus signs to imply that the form is a Boolean AND--all of the strings listed must be present to achieve the paste. For example, with d representing a delimiter, the format is

```
PASTE:dSTRING1d+....+dSTRING5d
```

For the OR form, the user lists strings and minus signs to imply that the form is a Boolean OR--any one of the listed strings need be present to achieve the paste. For example, with d representing a delimiter, the format is

```
PASTE:dSTRING1d-...-dSTRING5d
```

Note that these two special forms of the operand are equivalent in line or string mode.

## BUILD Command

The BUILD command enables the user to append additional text to his text file. When the user gives the BUILD command, the EDITOR subsystem positions the search pointer to the end of the file and responds with ENTER.

The text to be entered is typed on lines following the ENTER response. When text entry is complete, a carriage return only in response to the asterisk generates the - response.

The text entered following the ENTER response is appended to the file. When the carriage return is given to signal the end of text entry, the - response is given and the search pointer is returned to the beginning of the file.

```
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER, PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE SAME TERMINAL.

END OF FILE
BUILD
ENTER
*THE PROGRAM CAN BE CORRECTED OR CHANGED BY
*THE USER AS IF HE WERE CONVERSING BY PHONE.
*EXCEPT IN THIS CASE, THE CONVERSATION IS
*TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
*OF TERMINAL IN USE.
*(blank,carriage return)
*IF THE PROGRAM CONTAINS A MISTAKE, THE
*COMPUTER INFORMS THE USER.
*(carriage return)
-PRINT;*
TIME-SHARING PERMITS A DIALOGUE BETWEEN THE
COMPUTER AND USER, PERMITTING THE DIALOGUE
TO BEGIN IMMEDIATELY, WITHOUT WAITING FOR
THE COMPUTER TO COMPLETE PREVIOUS PROGRAMS.
DATA IS FED FROM THE TERMINAL DIRECTLY TO
THE COMPUTER AND ANSWERS ARE RECEIVED
QUICKLY AT THE SAME TERMINAL.
THE PROGRAM CAN BE CORRECTED OR CHANGED BY
THE USER AS IF HE WERE CONVERSING BY PHONE,
EXCEPT IN THIS CASE, THE CONVERSATION IS
TYPED OR DISPLAYED, DEPENDENT UPON THE TYPE
OF TERMINAL IN USE.
```

```
          IF THE PROGRAM CONTAINS A MISTAKE, THE
          COMPUTER INFORMS THE USER.

          END OF FILE
```

## RUNOFF Command

The RUNOFF command enables the user to access the RUNOFF subsystem from the EDITOR subsystem without the need to return to the SYSTEM ? level. When the user gives the RUNOFF command, the RUNOFF subsystem generates the "-" response to indicate its availability to accept a RUNOFF subsystem command. (See Section IV.) After the user has performed desired RUNOFF functions, a DONE command re-accesses the EDITOR subsystem.

## VERIFY Command and NOVERIFY Command

The VERIFY command enables the user to set the mode of the EDITOR subsystem so as to verify the execution of an EDITOR command. For positioning commands, the VERIFY command causes a printout of the line at which the search pointer is positioned when the positioning command is given. For text altering commands in line mode, the VERIFY command causes a printout of the line preceding the change, the affected change, and the line following the change. Although the line following the change is printed, the search pointer remains at the last line affected.

For text altering commands in string mode, the VERIFY command causes a printout of the one or more lines affected by the change. The NOVERIFY command removes the VERIFY mode.

```
          *EDITOR OLD filename
          -VERIFY
          (EDITOR positioning and text altering commands are verified
          by the EDITOR subsystem upon execution.  The VERIFY command
          will remain in effect until nullified by a NOVERIFY command.)
```

Verification of a particular EDITOR command is achieved by appending the letter V to the command verb.

```
          -FINDV:/xxx/;n        ≡ PS:/—/;n

          (Upon finding the nth occurrence of the specified line,
          the line is printed out.)

          -REPLACEVS:/xxx/:/bbb/

          (Upon replacement of string xxx by string bbb, the altered line
          is printed out.)
```

The appended V affects the command once only; the verification is not repeated for subsequent uses of the command.

## CASE Command and STANDARD Command

The CASE command enables the user to set the mode of the EDITOR subsystem so as to permit it to search a dual-case (upper and lower) text file from a terminal with a single-case keyboard.

The STANDARD command removes the EDITOR subsystem from the CASE mode.

The formats and execution of the CASE command are as follows, d representing a delimiter other than that used for delimiting string fields. The delimiter must be a character not contained within the file.

| Command | Execution |
|---|---|
| CASE | String fields of commands cause search and location by text, ignoring case. |
| CASE UPPER d | String fields of commands cause search and location of upper case text. The specified delimiter denotes upper case text for display or insertion. |
| CASE LOWER d | String fields of commands cause search and location of lower case text. The specified delimiter denotes lower case text for display or insertion. |

For example, a line of text in a file consists of the following:

```
EDITOR and RUNOFF are subsystems of TEXT EDITOR.
```

The user is restricted to an upper case terminal.

```
-CASE
-PRINT:/EDITOR/
EDITOR AND RUNOFF ARE SUBSYSTEMS OF TEXT EDITOR


-CASE UPPER $
-PRINT
$EDITOR$ AND $RUNOFF$ ARE SUBSYSTEMS OF $TEXT$ $EDITOR$

-REPLACES:/SUBSYSTEMS/:/$S$UBSYSTEMS/
```

The line of text in the file now consists of the following:

```
EDITOR and RUNOFF are Subsystems of TEXT EDITOR.
```

Note that the specified delimiter does not become part of the text.

## MODE Command

The MODE command provides the terminal operator with a method of determining previously established modes (Verify, String, Line, Case, etc.). The verb "MODE" or short form M can be typed to determine which modes have been set.

## TRANSPARENT Command

The TRANSPARENT (T) command allows the user to search the current file for all transparent characters (nonprinting characters octal 000 through 037). The search begins at the current line pointer position through to the end of file. Each line found containing transparent characters is printed and the character is bracketed by asterisks and printed in translated form. For example, a line containing a backspace would be printed as follows:

```
-T

This line has a backspace *BSP* here.
```

## MARK Command

Whenever a user types MARK, a search of the file is begun for a line commencing with a ".MARK" or ".MARK FILENAME". If a line starting with ".MARK" is located, and a file name is specified, the file will be accessed and the data on the specified file will replace the ".MARK" line. If the line does not contain a file name, the user will be queried as to the file to be accessed. If a ".MARK" line is not found, the user will be so informed.

Files accessed utilizing the ".MARK FILENAME" sequence may contain embedded ".MARK" lines. If the "MARK" command (verb) is followed with a repeat of ";*", each time a normal end of file condition is reached following a successful access of a specified "MARK" file, the current file is searched again to ensure that the accessed file did not contain a ".MARK" line.

Limitations:
1. MARK operates in a "NOVERIFY" environment. Whew!
2. The "MARK" command cannot be used in conjunction with the "LIMIT" function since "LIMIT" checks to see if file is line numbered.
3. Catalog/file strings are not permitted, nor are multiple files; e.g., file1;file2, etc.

## AFTLIN Command and BEFLIN Command

AFTLIN and BEFLIN are acronyms for "AFTer LINe" and "BEFore LINe". They allow the user to append data at the end of the line(s) or at the beginning of the line(s).

AFTLIN (short form A) command allows the user to append data to a line or number of lines specified in the repeat field. Input data can be entered in either of two forms.

1. Input data can follow the repeat factor (;n or ; *) in form ":/input data/", example:

   ```
   -A;1:/abcdef/
   ```

   where the string "abcdef" will be appended to the current line (;1). If the repeat factor (;n) is greater than one, the string "abcdef" will be appended to the current line plus the next n -1 lines.

2. Input data can follow the ENTER message.

   ```
   -A;1
   ENTER
   *abcdef
   *CR(carriage return)
   ```

   NOTE: In the above form, the numeric "1" specifies the current line. If more than the specified "n" lines of input is entered, the excess is ignored.

If ";*" appears in the repeat field, the input data is appended to all of the remaining lines. For example:

```
-A;*
ENTER
*001abc
*002def
*003ghi
*CR(carriage return)

END OF FILE - REQUEST EXECUTED nnn TIMES
```

In the above example, input line "001abc" is appended to the current line, input "002def" is appended to the current line plus one, and "003ghi" is appended to the current line plus two.

BEFLIN, is not permitted a short form since "B" would conflict with the verb BACKUP; the abbreviation BEFL is permitted. The same rules apply to BEFLIN as to the AFTLIN, except the input data is prefixed to the beginning of the line.

# SECTION IV

# RUNOFF SUBSYSTEM

The RUNOFF subsystem allows the user to print a text file in a previously determined format. The format is directed by control words entered in the file. The file must be built using the EDITOR subsystem. The RUNOFF control words can be entered during building of the file or inserted later during editing of the file.

In addition to imbedded control words, RUNOFF also uses commands that control the way in which the file is to be saved or printed. These commands are used after entering RUNOFF and are never inserted in the file.

Logging on the RUNOFF subsystem is the same procedure as that described in Section III for the EDITOR subsystem.

```
*RUNOFF
READY
```

Logging off the RUNOFF subsystem is accomplished by means of the Time Sharing System command BYE, given when the message RUNOFF COMPLETE appears or at the end of a page output when the system is expecting instructions as to how to proceed. If the user wishes to continue at the terminal instead of logging off at these times, he can give the REFORM or PRINT command (see below) with desired operands or he can give the Time Sharing System command DONE to return to the build mode level.

NOTE: RUNOFF subsystem availability is indicated by a READY response, in contrast to a "-" response for EDITOR subsystem availability.

## RUNOFF COMMANDS

The RUNOFF subsystem permits the use of the following commands:

```
REFORM
PRINT
SKIP n
NOSTOP
EDITOR
NUMBER
```

The REFORM command is required to format (remove RUNOFF control words entered during the building and/or editing process of) the file. The file can then be saved and/or printed. The PRINT command can be used to print a file formatted previously by the use of the REFORM command. The SKIP and NOSTOP commands can be used in conjunction with either the REFORM or PRINT commands. The EDITOR command can be given to access the EDITOR subsystem while in RUNOFF. The NUMBER command indicates the user has a line number file and desires to reformat the file without line numbers.

## REFORM Command

The operand of the REFORM command can contain four fields, separated by commas, and must contain at least two fields.

The first field specifies the file name of the data to be formatted. This field is required. If the file is a current file accessed by another time sharing subsystem, an asterisk can be used in the first field in lieu of the file name.

The second field specifies the file name under which the formatted data is to be saved. This field is optional, but must be present when the third field is not used.

The third field contains the command PRINT. This field is optional, but must be present when the second field is not used.

The fourth field contains the expression COUNT n. COUNT produces, in formatted text, the relative line number of the source file specified in the first field. N indicates the number of spaces set for the left margin of the formatted text. This fourth field is optional. Where the field PRINT is not used, the COUNT n field can replace it. The order of the fields can not be changed. The n portion of COUNT n is optional, with the following actions resulting:

1. If n is not present and RUNOFF does not encounter a left margin, six spaces are provided for the left margin.

2. If n is not present and RUNOFF does encounter a left margin setting, this margin setting is used.

3. If n is present and RUNOFF does not encounter a left margin setting, n designates the setting.

4. If n is present and RUNOFF does encounter a left margin setting, the setting is the larger of the two.

The following examples illustrate the use of the command (COUNT n can be used with any of the examples).

```
REFORM filename1,filename2,PRINT
READY
```

This form of the command causes file 1 to be formatted into pages and saved in file 2. At the same time, the formatted contents of file 2 are printed out at the terminal.

```
REFORM filename1,filename2
READY
```

This command formats file 1 into pages and saves the formatted text in file 2, to be printed out at a later time. (File 2 must be a previously defined file.)

```
REFORM filename1,,PRINT
READY
```

This command formats file 1 and transmits the formatted text to the terminal. (The contents of file 1 saved by EDITOR remain saved in unformatted form.)

## PRINT Command

The operand of the PRINT command contains only one field--the file name of a previously formatted text file. The file name must be the same as the second field of the REFORM command which saved the text.

```
PRINT filename2
READY
```

After a REFORM or PRINT command, the system types out READY. The file(s) specified are accessed but are yet to be acted upon. At this time, the commands SKIP n and NOSTOP n can be entered.

If printing is to be done, READY may be followed by a carriage return, or one or both of two commands--SKIP n and NOSTOP n.

If only a carriage return is used, the formatted text is printed out one page at a time, beginning at the first page of the file. After each page is complete, RUNOFF stops to allow the paper for the next page to be placed in the terminal. When the new paper is positioned, type a carriage return to start printing again. When all pages have been printed, RUNOFF COMPLETE is typed out.

```
REFORM filename1,,PRINT
READY
(carriage return)
POSITION PAPER NOW
(carriage return)
```

## SKIP n Command

The SKIP n command allows the user to obtain partial output of the file. Printing begins at page n+1. When printing stops at the end of each page, this command can be used.

```
PRINT filename2
READY
SKIP 8
READY
(carriage return)
POSITION PAPER NOW
(carriage return)
(The ninth page is printed out)
SKIP 3
READY
(carriage return)
(The thirteenth page is printed out)
```

## NOSTOP Command

The NOSTOP command can be used when the terminal is loaded with continuous paper. RUNOFF does not stop after each page is printed. The SKIP n command can be used with NOSTOP. The form NOSTOP n permits n consecutive pages to be printed before a stop is made at the end of the nth page.

```
PRINT filename2
READY
SKIP 8
READY
NOSTOP
READY
(carriage return)
POSITION PAPER NOW
(carriage return)
(Printing begins at the ninth page and continues
to the end of the file unless stopped
manually at the terminal.)
```

## NUMBER Command

The NUMBER command indicates the user has a line numbered and desires to reformat the file without line numbers. The usage of the command is the same as for the SKIP and NOSTOP commands.

## EDITOR Command

The EDITOR command can be used to access the EDITOR subsystem while in RUNOFF subsystem without the need to return to the subsystem selection level. Upon being given the EDITOR command, the EDITOR subsystem responds with the "-" response. The user can then perform desired editing function and return to the RUNOFF subsystem by means of the Time Sharing System command DONE.

A current file must have been created if the EDITOR command is to be used while in the RUNOFF subsystem. If the system selected at log-on time is RUNOFF and no current file exists, the use of the EDITOR command generates the message

```
<52> CURRENT FILE NOT DEFINED
```

## RUNOFF FORMAT CONTROL WORDS

The RUNOFF format control words which can be entered in the text file during the building or editing process are listed below. Each of these can be used in an abbreviated form, utilizing the first four letters (e.g., allc).

```
.allcaps n
.beginpage n
.boldface n
.bottommargin n
.break
.center n
.comment
.doublespace
.fill
.footing x,n
.header x,n
.ignore x,x
.indent n
.justify
.leftdent n
.linelength n
.literal
.margin t,b,l,r
.multispace n
.nodent
.nofill
.nojust
.notab
.page x,y,n
.paperlength n
.paragraph
.point n
.reference (x...x)
.replace x,x
.scoreunder n
.singlespace
.space n
.subheading x,n
.subfooting x,n
.subparagraph n
.tabulate t,n,,,n
.topmargin
.undent n
```

The following rules apply to use of RUNOFF format control words:

1. Each control word must be preceded by a period and followed by a carriage return. Any text material typed on the same line as the control word is ignored when printing out the formatted text.

2. Control words can be typed in either uppercase or lowercase.

3. All legitimate control words are ignored when printing out and do not appear in the text.

4. Control words that are to remain in effect throughout the file can be entered once at the beginning of the file and need not be repeated unless they are cancelled by an imbedded control word. For example:

```
.PAPE 66
.LINE 60
.TOPM 6
.BOTT 6
.SING
.FILL
.JUST
```

5. The control words and values shown in the above example are those preset by RUNOFF and need not be entered; they remain in effect unless changed by the user. No page numbering occurs unless .PAGE is encountered. Care should be exercised in specifying page size parameters. RUNOFF formats a full page before printing. The following page matrix formula can be used to determine a large page format. Exceeding the results of this calculation leads to a memory fault.

```
4P + (P-T-B+2)(L+2) = 7000
where P = .paperlength n
      T = .topmargin n
      B = .bottommargin n
      L = .linelength n
```

6. Words should not be hyphenated at the end of a line when using .FILL. The carriage return following the hyphen is treated as a space character and the hyphenated word could appear in the middle of a line of text as follows:

   MULTI- PLIED

   A compound, such as "right-hand", is treated as one word by RUNOFF and is not split over two lines in order to fill or justify lines.

.ALLCAPS n

Print next n lines in upper case. If n is not used, only the next line is printed in upper case.

.BEGINPAGE n

Place text following control word on a new page. If n is specified, the new page is numbered n and succeeding pages are referenced by n.

.BOLDFACE n

Overprint the next n lines. If n is not used, only the next line is overprinted. The use of .BOLDFACE and .SCOREUNDER on the same line(s) results in .SCOREUNDER operation only.

## .BOTTOMMARGIN n

Specify the space from the last line of text output to the bottom of the paper. N should equal the number of lines desired. If this control word is not used, RUNOFF presets the margin to 6. Page numbers, if requested, are printed within the margin space.

## .BREAK

End previous line and start a new line, without inserting a blank line. The lines previous and following the use of this control word are not joined even though .FILL has been specified.

## .CENTER n

Center the next n lines. When n is not used, only the next line is centered. When centering, do not include any other RUNOFF control words within the lines to be centered.

## .COMMENT

Prevent printing of all lines of text until another RUNOFF control word is encountered.

## .DOUBLESPACE

Specify text to be printed out double spaced.

## .FILL

Lengthen short lines by moving words from the following line and shorten long lines by moving words to the following line. This is preset by RUNOFF and is in effect until a .NOFIL is encountered. .FILL does not insert spaces to justify the right-hand margin.

.FOOTING x,n

Specify the number of lines and the position of the foot line to be printed at the bottom of a page. One line space is automatically inserted before the footing.

N indicates the number of lines. X can be one of the following:

C - Centered on each page.
R - Right justified on each page.
L - Left justified on each page.
A - Alternately right justified on odd numbered pages,
    left justified on even numbered pages.
E - Left justified on even numbered pages.
O - Right justified on odd numbered pages.

The .FOOTING control word can be entered only at the beginning of the file or after .BEGINPAGE within the file if the foot line is to be changed. Termination of foot lines is accomplished by use of .FOOTING NO or .FOOTING 0 (numeric).

.HEADER x,n

Specify the number of lines and the position of the header to be printed on a page. One line space is automatically inserted after the header. To insert a blank line in the header, use a space character before the carriage return. N indicates the number of lines. X can be one of the following:

C - Centered on each page.
R - Right justified on each page.
L - Left justified on each page.
A - Alternately right justified on odd numbered pages,
    left justified on even numbered pages.
E - Left justified on even numbered pages.
O - Right justified on odd numbered pages.

For example:

```
*.HEADER R,3
*TIME-SHARING
*(space)
*(space)
*
```

The .HEADER control word can be entered at the beginning of the file and also just before or after .BEGINPAGE within the file if the heading is to be changed. Termination of header lines is accomplished by use of .HEADER NO or .HEADER 0 (numeric).

.IGNORE x,x,.....

Prevent the symbols listed in the operand from being used as text characters. Up to 16 characters may be listed for suppression. Use of the characters as text is resumed by .IGNORE NO or .IGNORE 0. Numerics are not valid symbols for use with .IGNORE.

.INDENT n

Indent each following line of text the number of spaces specified. Indentation is preset to zero and is accumulative; that is, subsequent .INDENT control words add to the total indentation until a .NODENT, .LEFTDENT, or .UNDENT is encountered.

.JUSTIFY

Insert spaces into the line between words to justify the right-hand margin to the length specified by .LINE. This is preset by RUNOFF and remains in effect until a .NOJUST is encountered.

.LEFTDENT n

In an indented area, subtract n spaces from the total indentation, for all following lines until an .INDENT, .NODENT, or .UNDENT is encountered. If n is greater than the total indentation, the total is set to zero.

.LINELENGTH n

Specify the length of the line, in characters, for filling and justifying. N should equal the length in inches multiplied by 10. (6-inch line = 60, 7-inch line = 70, etc.) If this control word is not used, RUNOFF presets the line to 60. The left margin position on the paper is determined manually at the terminal.

.LITERAL

Print a RUNOFF control word when it appears as part of the text. .LITE can be used on the same line, preceding the control word, or on the line before the control word as shown below.

```
.LITERAL
.LITE can be used on the same line,
.LITERAL .PAGE n starts page numbering.
```

.MARGIN T,B,L,R

Set the four margins of a page. The numerics for T(top) and B(bottom) set the line count for the top and bottom margins. Numerics for L(left) and R(right) set the character counts for left and right margins. T (top) and B (bottom) margins must be specified. Nulling of relative fields will result in a top and bottom margin of zero.

NOTE: The .MARGIN control word cannot be utilized to change top or bottom margins in the middle of a page. To change top or bottom margins on a succeeding page, use .MARGIN within the bounds of the page, immediately following .BEGINPAGE or the page break.

.MULTISPACE n

Specify text is to be printed out with n line spaces between text lines. This command overrides any .SINGLESPACE or .DOUBLESPACE command.

.NODENT

In an indented area, reset the total indentation to zero.

.NOFILL

Print all lines exactly as they were typed into the file.

.NOJUST

Stop justification.

.NOTAB

Stop tabulation and return to the previous format instructions.

.PAGE x,y,n

Start page numbering. If n is not present, numbering begins with page 1. If page numbers are to start with any other number, n should equal the starting page number.

x and y specify where page numbers are to appear. x and y can take one or more of the following values:

B - Bottom of page
T - Top of page
C - Center
L - Left justified
R - Right justified
A - Alternating (odd numbers on the right, even on the left)

Page numbers, if requested, are inserted within the specified margin.

The example below would cause numbering to begin with page 1, numbers to be printed on alternate sides of the pages, at the bottom.

```
.PAGE B,A,1
```

If _n_ is specified and _x_ and _y_ are not, page numbers appear centered and at the bottom of the page.

.PAPERLENGTH n

Specify the total length of the paper. _n_ should equal the length in inches multiplied by 6. (11-inch paper = 66, 14-inch paper = 84, etc.) If this control word is not used, RUNOFF presets the length to 66.

.PARAGRAPH

Preset the line length to its specification before the previous .SUBP control word. In an indented region, the former indentation total regains control.

.PARAGRAPH n1,n2

The n1 field causes the left margin to be indented the number of spaces specified. The n2 field causes the first line of the paragraph to be indented the number of spaces specified.

.POINT n

Cause a new page to be formatted. The page number is not incremented, but appears with a period followed by 1 (p.1). The page incrementing continues behind the period until terminated with the control word .BEGIN, when page _p_ resumes incrementing. If the operand _n_ is used, the 1 following the period is replaced with _n_.

.REFERENCE (x...x)

This causes the text within the parentheses to be printed as a footnote at the bottom of the same page. The .REFE must be preceded by a footnote indicator that must also be the first character(s) within the parentheses; i.e.; no space is permitted between the first parenthesis and the indicator.

```
- - - - - - - - - - - - - - - - - - - - - -        see
(1)
.REFE ((1) This text is a footnote  printed  at  the
bottom of the page.) below.- - - - - - - - - - - - -
-
```

When printed by RUNOFF, appears as

```
          - - - - - - - - - - - - - - - - - - - - -  see (1)
          below.- - - - - - - - - -
          - - - - - - - - - - - - - - - - - - - - - - - -  -
          - - - - - - - - - - - - - - - - - - - - - - - -  -
          - - - - - - - - - - - - -

          _____________________

          (1) This text is a footnote printed at the bottom
          of the page.
```

.REPLACE x,x,......

Cause the symbols listed in the operand to be replaced with space characters. The space characters supplied are not used as word string terminators in formatting the text. This enables the user to reserve character spaces for special character insertion, superscripting, subscripting, etc. Up to 16 symbols can be listed for replacement. Use of the symbols as text is resumed by the control word .REPLACE NO or .REPLACE 0 (numeric).

.SCOREUNDER n

Cause the next input text line or each of the next n input text lines to be underscored (underlined) in the formatted text. If n is omitted, underscoring is performed only on the next line of text. The use of .BOLDFACE and .SCOREUNDER on the same line(s) results in .SCOREUNDER operation only.

.SINGLESPACE

Specify text to be printed out single spaced. If this control word is not used, and if .DOUBLESPACE is not specified, RUNOFF presets the format to single space.

.SPACE n

Insert n blank lines spaces. If the end of the page is reached before n (spaces) are provided, spacing stops. Blank lines are not carried over to the next page.

## .SUBHEADING x,n

Specify the number of lines to be printed as a subheading to a previously defined header.

N indicates the number of lines. X can be one of the following:

C - Centered on each page.
R - Right justified on each page.
L - Left justified on each page.
A - Alternately justified on odd numbered pages,
    left justified on even numbered pages.
E - Left justified on even numbered pages.
O - Right justified on odd numbered pages.

The .SUBHEADING control word can be changed after a .BEGINPAGE within the file. Termination of the use of the subheading is accomplished using .SUBHEADING NO or .SUBHEADING 0 (numeric).

## .SUBFOOTING x,n

Specify the number of lines to be printed as a subfooting to a footing previously defined.

N indicates the number of lines. X can be one of the following:

C - Centered on each page.
R - Right justified on each page.
L - Left justified on each page.
A - Alternately right justified on odd numbered pages,
    left justified on even numbered pages.
E - Left justified on even numbered pages.
O - Right justified on odd numbered pages.

The .SUBFOOTING control word can be entered only at the beginning of the file or after a .BEGINPAGE within the file. Termination of the use of the subfooting is accomplished using .SUBFOOTING NO or .SUBFOOTING 0 (numeric).

## .SUBPARAGRAPH n

Indent the beginning of each line n and subtract n spaces from the end of the line. For example, if the line length is 60 and .SUBP 5 is used, the lines following are 50 characters long.

In an indented region, the subparagraphing is affected by the total indentation. For example:

```
.LINE 60
.INDENT 5
.SUBP 5
```

results in lines 45 characters long, indented 10 spaces from the left margin.

## .TABULATE t,n,...n

Set simulated tabs on the horizontal line locations specified by the values of n. When building the file, enter a tabulation character (any keyboard character other than a blank, control character, or one being used as a delimiter) at the beginning of each tabulated field, as this character is used by RUNOFF. See the following examples.

```
.TABU  t,10,20,30
txxxxtyyyytzzzz
txxxxtyyyytzzzz
```

When printing in RUNOFF, the following result

```
(columns) 10           20           30
           xxxx         yyyy         zzzz
           xxxx         yyyy         zzzz
```

When using a terminal that has no tab control key, any symbol can be chosen as a tabulation character. The symbol is not printed out during RUNOFF but can be read when using EDITOR. .TABU operates in a .NOFIL environment.

## .TOPMARGIN n

Specify the space from the top of the paper to the first line of output. N should equal the space desired multiplied by 6. (1-inch margin = 6, etc.) If this control word is not used, RUNOFF presets the margin to 6. Page numbers, if requested, are printed within the margin space.

## .UNDENT n

In an indented area, causes n to be subtracted from the total indentation for the next line only.

# APPENDIX A

## SUMMARY OF EDITOR COMMANDS

In this summary, the following conventions are used:

| | |
|---|---|
| xxx | - represents a string of characters, any length, that matches the beginning of one or more lines in the file. |
| yyy and zzz | - represent string of characters, any length, that may or may not match the beginning of a line in the file. |
| bbb | - represents a string of characters to be inserted or to replace another string. |
| n | - represents a number or an asterisk. The * indicates the action is to be repeated throughout the file. |
| / | - slants are used as delimiters; any other character can be used instead. |
| _ | - the underscored initial letter of the command implies permissible abbreviation. If the command contains an s for string mode, the abbreviation can be the initial and final letters. |

| Command | Execution |
|---|---|
| AFTLIN;n:/data/ | Append the "data" specified to the end of n lines. |
| BACKUP;n | Return search pointer to the beginning of the file or back up the number of lines specified. |
| BEFLIN;n:/data/ | Append the "data" specified to the beginning of n lines. |
| BUILD | Append text to file. |
| CASE | Set mode of EDITOR to permit it to search dual-case text. |
| COPY;n | Copy line or lines where search pointer is currently located. |
| COPY:/xxx/ | Copy line identified by xxx. |
| COPY:/xxx/;n | Copy the next n lines identified by xxx. |
| COPY:/xxx/,/yyy/ | Copy the lines identified by xxx through yyy. |

| | |
|---|---|
| COPY:/xxx/,/yyy/;n | Copy the next n line groups identified by xxx through yyy. |
| COPYS:/yyy/ | Copy the line that contains the specified string. |
| COPYS:/yyy/;n | Copy n occurrences of the line that contains the specified string. |
| COPYS:/yyy/,/zzz/ | Copy text between points yyy and zzz, inclusive. (A repeat field can be used with this form.) |
| COPY:/stl/+.../stn/ | Copy line containing all specified strings. (A repeat field can be used with this form.) |
| COPY:/stl/-.../stn/ | Copy line containing only one of specified strings. (A repeat field can be used with this form.) |
| CUT or CUT;n | Copy and remove line or lines where search pointer is currently located. |
| CUT:/xxx/ | Copy and remove line identified by xxx. |
| CUT:/xxx/;n | Copy and remove the next n lines identified by xxx. |
| CUT:/xxx/,/yyy/ | Copy and remove the block of lines identified by xxx through yyy. |
| CUT:/xxx/,/yyy/;n | Copy and remove the next n blocks of lines identified by xxx through yyy. |
| CUTS:/yyy/ | Copy and remove the line that contains the specified string. |
| CUTS:/yyy/;n | Copy and remove the next n occurrences of the line that contains the specified string. |
| CUTS:/yyy/,/zzz/ | Copy and remove text between points yyy and zzz, inclusive. (A repeat field can be used with this form.) |
| CUT:/stl/+.../stn/ | Copy and remove line containing all specified strings. (A repeat field can be used with this form.) |
| CUT:/stl/-.../stn/ | Copy and remove line containing any one of specified strings. (A repeat field can be used with this form.) |
| DELETE or DELETE;n | Delete line or lines where search pointer is currently located. |
| DELETE:/xxx/ | Delete line identified by xxx. |
| DELETE:/xxx/;n | Delete the next n lines identified by xxx. |
| DELETE:/xxx/,/yyy/ | Delete the block of lines identified by xxx through yyy. |
| DELETE:/xxx/,/yyy/;n | Delete the next n blocks of lines identified by xxx through yyy. |
| DELETES:/yyy/ | Delete specified string. |

| | |
|---|---|
| DELETES:/yyy/;n | Delete n occurrences of specified string. |
| DELETES:/yyy/,/zzz/ | Delete text between points yyy and zzz, inclusive. (A repeat field can be used with this form.) |
| DELETE:/stl/+.../stn/ | Delete line containing all specified strings. (A repeat field can be used with this form.) |
| DELETE:/stl/-.../stn/ | Delete line containing any one of specified strings. (A repeat field can be used with this form.) |
| FIND | Advance search pointer one line. |
| FIND;n | Advance search pointer n lines. |
| FIND:/xxx/;n | Find nth line identified by xxx. |
| FINDS:/yyy/ | Find line containing specified string. |
| FINDS:/yyy/;n | Find line containing the nth occurrence of specified string. |
| INSERT | Insert after the line where the search pointer is currently located, except when at the beginning of the file. |
| INSERT:/xxx/ | Insert after the line identified by xxx. |
| INSERT:/xxx/;n | Insert after each of the next n lines identified by xxx. |
| INSERTS:/yyy/ | Insert after point yyy. |
| INSERTS:/yyy/;n | Insert after each of n successive occurrences of yyy. |
| INSERT:/xxx/:/bbb/ | Insert string bbb after the line identified by xxx. |
| INSERT:/xxx/;n:/bbb/ | Insert string bbb after each of the next n lines identified by xxx. * |
| INSERTS:/yyy/:/bbb/ | Insert string bbb after yyy. |
| INSERTS:/yyy/;n:/bbb/ | Insert string bbb after each of n successive occurrences of yyy. * |
| INSERT:/stl/+.../stn/ | Insert after line containing all specified strings. (A repeat field can be used with this form.) |
| INSERT:/stl/-.../stn/ | Insert after line containing any one of specified strings. (A repeat field can be used with this form.) |
| LINE | Return EDITOR to line mode. |
| MARK filename | Search the file for a .MARK line. |
| MODE | Used to determine previously established modes (Verify, String, Line, Case, etc.) |

| | |
|---|---|
| NOVERIFY | Nullify VERIFY command. |
| PASTE | Insert text after the line where the search pointer is located, except when at the beginning of file. |
| PASTE:/xxx/ | Insert text after the line identified by xxx. |
| PASTE:/xxx/;n | Insert text after the next n lines identified by xxx. |
| PASTES:/yyy/ | Insert text after point yyy. |
| PASTES:/yyy/;n | Insert text after each of n successive occurrences of yyy. |
| PRINT | Print one line. |
| PRINT;n | Print n consecutive lines. |
| PRINT;* | Print entire file. |
| PRINT:/xxx/ | Print line identified by xxx. |
| PRINT:/xxx/;n | Print the next n lines identified by xxx. |
| PRINT:/xxx/,/yyy/ | Print the block of lines starting with xxx through yyy. |
| PRINT:/xxx/,/yyy/;n | Print the next n blocks of lines starting with xxx through yyy. |
| PRINTS:/yyy/ | Print line containing the specified string. |
| PRINTS:/yyy/;n | Print n lines containing occurrences of the specified string. |
| PRINTS:/yyy/,/zzz/ | Print from line containing yyy to line containing zzz, inclusive. (A repeat field can be used with this form.) |
| PRINT:/stl/+.../stn/ | Print line containing all specified strings. (A repeat field can be used with this form.) |
| PRINT:/stl/-.../stn/ | Print line containing any one of specified strings. (A repeat field can be used with this form.) |
| * REPLACE | Replace line where search pointer is currently located. |
| REPLACE:/xxx/ | Replace line identified by xxx. |
| REPLACE:/xxx/;n | Replace the next n lines identified by xxx. |
| REPLACE:/xxx/,/yyy/ | Replace the block of lines starting with xxx through yyy. |
| REPLACE:/xxx/,/yyy/;n | Replace the next n blocks of lines starting with xxx through yyy. |
| REPLACES:/yyy/;n | Replace n successive occurrences of yyy. |

| | |
|---|---|
| REPLACES:/yyy/,/zzz/ | Replace text between points yyy and zzz, inclusive. (A repeat field can be used with this form.) |
| REPLACE:/xxx/:/bbb/ | Replace line identified by xxx with string bbb. |
| REPLACE:/xxx/;n:/bbb/ | Replace the next n lines identified by xxx with bbb. |
| REPLACES:/yyy/:/bbb/ | Replace string yyy with bbb. |
| REPLACES:/yyy/;n:/bbb/ | Replace n successive occurrences of yyy with bbb. |
| REPLACES:/yyy/,/zzz/:/bbb/ | Replace text between points yyy and zzz, inclusive, with string bbb. (A repeat field can be used with this form.) |
| REPLACE:/stl/+.../stn/ | Replace line containing all specified strings. (A repeat field can be used with this form.) |
| REPLACE:/stl/-.../stn/ | Replace line containing any one of specified strings. (A repeat field can be used with this form.) |
| RUNOFF | Access RUNOFF subsystem without need to return to system selection level. |
| STANDARD | Nullify CASE command. |
| STRING | Place EDITOR in string mode. |
| TRANSPARENT | Search the current file and print all lines containing transparent characters (nonprinting characters octal 000 through 037). |
| VERIFY | Set mode of EDITOR to verify execution of EDITOR commands. |

# APPENDIX B

## SUMMARY OF RUNOFF COMMANDS AND CONTROL WORDS

In this summary, the following conventions are used:

| | |
|---|---|
| n | represents any number |
| F1 | represents an input file |
| F2 | represents the name of a formatted file |

| Command | Execution |
|---|---|
| REFORM F1,F2 | Format the contents of F1 and save it under the F2 name. |
| REFORM F1,,PRINT | Print the formatted contents of F1 but do not save the file in formatted form. |
| REFORM F1,F2,PRINT | Format the contents of F1, save it under the F2 name, and print the formatted text. |
| PRINT F2 | Print the formatted file saved by a previous REFORM command. |
| SKIP n | Skip the specified number of formatted pages. Printing starts on page n+1. |
| NOSTOP | Print continuously, without stopping at the end of each page. |
| EDITOR | Access the EDITOR subsystem without need to return to SYSTEM? level. |

| Control Words | Execution |
|---|---|
| .ALLCAP n | Print next n lines in upper case. |
| .BEGINPAGE n | End printing on this page and place following text on the next page, numbered n. |
| .BOLDFACE n | Overprint the next n lines. |
| .BOTTOMMARGIN n | Specify the size of the bottom margin in terms of print lines. |
| .BREAK | Do not join the following line to the previous line (as in .FILL or .JUST). |
| .CENTER n | Center the next n lines. |

| | |
|---|---|
| .COMMENT | Do not print following lines of text until another control word is found. |
| .DOUBLESPACE | Double space the formatted text. |
| .FILL | Move words to shorten long lines and lengthen short lines. |
| .FOOTING x,n | Print n lines of foot line in location specified. |
| .HEADER x,n | Specify the number of lines and location of the page heading. That number of following lines will be printed at the top of each page. |
| .IGNORE x,x | Cause symbols listed not to be used as text characters. |
| .INDENT n | Indent n spaces at the beginning of all following lines. N is added to the accumulated total. |
| .JUSTIFY | Insert spaces between words to justify the right margin. |
| .LEFTDENT n | Subtract n spaces from the accumulated indent total for all following lines. |
| .LINELENGTH n | Specify the length of the line in terms of character spaces -- 10 per inch. |
| .LITERAL | Print the following RUNOFF control word as part of the text. |
| .MARGIN t,b,l,r | Set designated margins. |
| .MULTISPACE n | N space the formatted text. |
| .NODENT | Set the accumulated total of indent n's to zero. |
| .NOFILL | Print all lines as they were entered. |
| .NOJUST | Do not justify right margin. |
| .NOTAB | Stop tabulation and return to previous format. |
| .PAGE x,y,n | Number pages, beginning with n. Print page numbers in location x,y. |
| .PAPERLENGTH n | Specify the size of the paper in terms of print lines -- 6 per inch. |
| .PARAGRAPH | After subparagraphing, return to the previous line length. |
| .PARAGRAPH n1, n2 | The n1 field causes the left margin to be indented the number of spaces specified. The n2 field causes the first line of the paragraph to be indented the number of spaces specified. |
| .POINT n | Cause a new page to be formatted. |
| .REFERENCE (x..x) | Print footnote within the parentheses at the bottom of the page. |

|  |  |
|---|---|
| .REPLACE x,x | Listed symbols are replaced with space characters. |
| .SCOREUNDER n | Underscore next n lines. |
| .SINGLESPACE | Single space the formatted text. |
| .SPACE n | Insert n line spaces before printing next lines. |
| .SUBFOOTING x,n | Print n lines of subfoot lines in location specified. |
| .SUBHEADING x,n | Print n lines of subhead line in location specified. |
| .SUBPARAGRAPH n | Shorten lines by n character spaces at the beginning and end of each line. |
| .TABULATE t,n,...,n | Set tabs as specified by n. |
| .TOPMARGIN n | Specifies the size of the top margin in terms of print lines. |
| .UNDENT n | Subtract n spaces from the indent total for the next line only. |

# APPENDIX C

# RUNOFF EXAMPLES

Examples are given on the following pages to illustrate the use of RUNOFF. The left-hand page contains the text and instructions in the file. The right-hand page shows the same portion of the file as it is formatted by RUNOFF.

```
.pape 65
.line 67
.page 1
.topm 6
.bott 6
.just
.repl &
.header r,l
Text Editor
.subheading r,l
Examples
.space 4
.cent
SECTION I
.space 2
.cent
INTRODUCTION
.space 4
&&&&&This manual describes the Text-Editing Subsystems of the
Time Sharing System, EDITOR and RUNOFF. Use of these subsystems
does not require any knowledge of programming; however, the
following brief descriptions of computer systems and the terms
used in the manual will be helpful to the terminal operator.
.space 2
&&&&&In this manual, a "computer system" is an information
processing system. It may be located many miles from the
terminal through which information is being entered. The total
system consists of hardware (printers, card readers and punches,
permanent magnetic storage devices, processing equipment, etc.)
and programs (sets of instructions that tell a computer how to
accomplish a specific task). The Time Sharing System is one of
many such programs.
.space 2
&&&&&The Time Sharing System is made up of several small
programs called "subsystems". (See Figure 2 and (1) below.)
This manual covers two of these subsystems in detail. Other
subsystems, not required for text-editing purposes, are covered
in other manuals.
.nojust
.refe ((1) See Text Editor manual, DD18.)
.begi
```

## SECTION I

## INTRODUCTION

This manual describes the Text-Editing Subsystems of the Time Sharing System, EDITOR and RUNOFF. Use of these subsystems does not require any knowledge of programming; however, the following brief descriptions of computer systems and the terms used in the manual will be helpful to the terminal operator.

In this manual, a "computer system" is an information processing system. It may be located many miles from the terminal through which information is being entered. The total system consists of hardware (printers, card readers and punches, permanent magnetic storage devices, processing equipment, etc.) and programs (sets of instructions that tell a computer how to accomplish a specific task). The Time Sharing System is one of many such programs.

The Time Sharing System is made up of several small programs called "subsystems". (See Figure 2 and (1) below.) This manual covers two of these subsystems in detail. Other subsystems, not required for text-editing purposes, are covered in other manuals.

(1) See Text Editor manual, DD18.

```
.pape 65
.line 67
.repl &
.header r,1
Text Editor
.subheading r,1
Examples
.footing c,1
Time Sharing System
.subfooting c,2
Text
Editor
.space 4
.cent
PROCESSOR
.space 3
.nofil
.cent 4
Memory
Where Programs
are Stored
(During Use)
.space 3
.tabu z,10,29,44
zMagnetic Tapes
zDisks, and DrumszInput/OutputzPrinters
zwhere programszControllerszCard Punches
zare stored whenzzCard Readers
znot being used
.notab
.justify
.space 3
.cent
TERMINAL(S)
.space 3
.cent
Figure 1-1. Information Processing System
.space 4
&&&&The following verbs may not have an operand field:
.space 2
.tabu t,10,31,50
tLINE or LtRUNOFFtSTANDARD
tSTRING or StVERIFY
tBUILDtNOVERIFY
.notab
.fill
.begi
```

PROCESSOR

Memory
Where Programs
are Stored
(During Use)

| | | |
|---|---|---|
| Magnetic Tapes, Disks, and Drums where programs are stored when not being used | Input/Output Controllers | Printers<br>Card Punches<br>Card Readers |

TERMINAL(S)

Figure 1-1. Information Processing System

The following verbs may not have an operand field:

| | | |
|---|---|---|
| LINE or L | RUNOFF | STANDARD |
| STRING or S | VERIFY | |
| BUILD | NOVERIFY | |

Time Sharing System
Text
Editor

```
.pape 65
.line 67
.repl &
.header r,l
Text Editor
.subheading r,l
Examples
.space 4
&&&&&The use of the verbs and operands are fully explained
and illustrated in
.score
Editor Commands
later in this chapter. The restrictions and usage rules
which apply to the operand field are explained in
.score
Operand Field
below.
.space 2
.subp 5
The editor responds to the commands with messages that
inform the user when a command has been executed, a mistake
in command format has been made, or the end of the file has
been reached. These messages are described in
.score
Responses from Editor.
.para
.space 3
.allcaps
operand field
.space 2
&&&&&As stated above, the operand field can contain one
or more of the following:
.space 2
.indent 10
.undent 5
1.   Mode Indicators -
"S" for string mode and "L" for line mode
.space
.undent 5
2.   String field, preceded by a colon
.space
.undent 5
3.   Repeat field, preceded by a semicolon
.leftdent 5
.space 2
If more than one of these items is used in a single
command, the order must be as shown previously.
.nodent
.space 3
.score
Mode Indicators
.space 2
&&&&&The mode indicators used with the Editor verbs are
"S" for string mode and "L" for line mode. The mode
determines the type of operation to be performed and the
interpretation of the string field. See Figure 3.
.ignore no
.begin
```

The use of the verbs and operands are fully explained and illustrated in Editor Commands later in this chapter. The restrictions and usage rules which apply to the operand field are explained in Operand Field below.

The editor responds to the commands with messages that inform the user when a command has been executed, a mistake in command format has been made, or the end of the file has been reached. These messages are described in Responses from Editor.

## OPERAND FIELD

As stated above, the operand field can contain one or more of the following:

1. Mode Indicators -
   "S" for string mode and "L" for line mode
2. String field, preceded by a colon
3. Repeat field, preceded by a semicolon

If more than one of these items is used in a single command, the order must be as shown previously.

### Mode Indicators

The mode indicators used with the Editor verbs are "S" for string mode and "L" for line mode. The mode determines the type of operation to be performed and the interpretation of the string field. See Figure 3.

## INDEX